EasyChair Preprint

№ 10376

GoatPose: a Lightweight and Efficient Network with Attention Mechanism

Yaxuan Sun, Annan Wang and Shengxi Wu

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

June 11, 2023

GoatPose: A Lightweight and Efficient Network with Attention Mechanism

Yaxuan Sun[1,2][0009−0000−3149−4654], Annan Wang[1][0009−0002−1833−0617], and Shengxi Wu[1,3][0000−0002−1647−6586]

[1] East China University of Science and Technology, Shanghai, China
[2] First author, School of information science and engineering, ECUST, Shanghai
`{blade220284}@gmail.com`
[3] Corresponding Author, School of information science and engineering, ECUST, Shanghai
`{wushengxi}@ecust.edu.cn`

Abstract. Keypoint detection is an essential part of human pose estimation. However, due to resource constraint, it's still a challenge to deploy complex convolutional networks to edge devices. In this paper, we present GoatPose: a lightweight deep convolutional model for real-time human keypoint detection incorporating attention mechanism. Since the high computational cost is associated with the frequently-use convolution block, we substitute it with LiteConv block, which conducts cheap linear operation to generate rich feature maps from the intrinsic features with low cost. This method significantly accelerates the model while inevitablely loses a part of spatial information. To compensate for the information loss, we introduce NAM attention mechanism. By applying channel weighting, the model can focus more on the important features and enhance the feature representation. Results on the COCO dataset show the superiority of our model. With the complexity of our model reduced by half and the computational speed doubled, the accuracy of our model is basically the same as that of the backbone model. We further deploy our model on NVIDIA Jetson TX2 to validate its real-time performance, indicating that our model is capable of being deployed and widely adopted in real-world scenarios.

Keywords: Human key point detection · Lightweight network · High-resolution representation · Attention mechanism · Model deployment

1 Introduction

Human pose estimation aims to accurately detect the key points of the human body such as the head, shoulders, elbows, wrists, knees, and ankles. The detection task has broad applications, including motion capture, human-computer interaction, pedestrian tracking, etc. Deep learning has been successfully applied in the human body key point detection task, however, it's still a challenge to deploy complex networks to edge devices due to resource constraints.

Lightweight model are needed for human key point detection tasks in practical application. One popular approach to designing lightweight networks is to borrow techniques such as depth-wise separable convolution and channel shuffling from classification networks [1–3] to reduce computational redundancy. Model compression is also a common approach to generate lightweight model [4–9]. In this paper, we present a lightweight high-resolution network GoatPose.

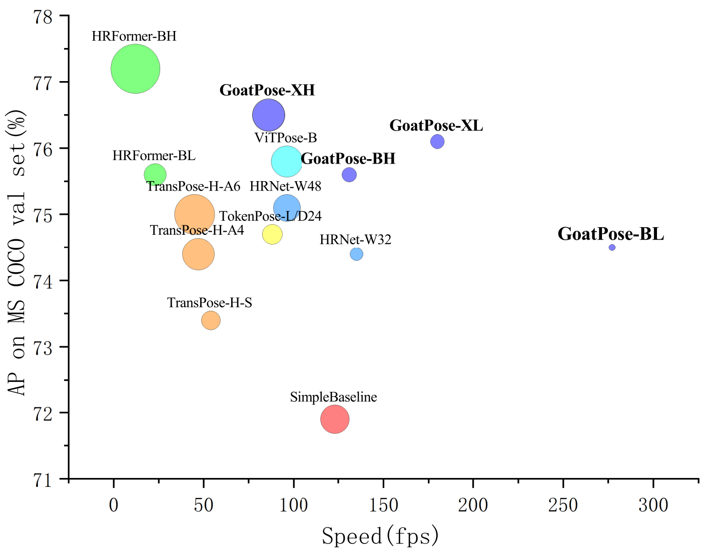


Fig. 1: AP and Speed on MSCOCO val set

Using high-resolution network HRNet[10] as our backbone, GoatPose maintains high-resolution representations through the whole process,therefore obtaining strong semantic information. Considering that the frequently-used convolutions excessively consume computational resources, we redesign the convolution block, incorporating cheap linear operation into it, which can generate rich feature maps from the intrinsic features with cheap cost[11, 12]. To compensate for the information loss caused by linear operation, we introduce the attention mechanism. By weighting the channels, the model can adaptively find areas that need attention and highlight important features while suppressing irrelevant features[13–15]. Our method on the COCO keypoint detection dataset[16] demonstrates superior results, as shown in **Figure 1**. We further deploy our model on NVIDIA Jetson TX2 to validate the real-time performance. It turns out that our model has a high AP of 74.5% with only 3.48 GFLOPs, outperforming prior state-of-the-art efficient pose estimation models. We believe our work will push the frontier of real-time pose estimation on edge.

Our main contributions include:

- We propose a lightweight network GoatPose that generates highresolution feature maps with low cost. The key of the model lies in the introduction of cheap operation and attention mechanism.
- We demonstrate the effectiveness of GoatPose on the COCO dataset. Our model outperforms all other models and achieves excellent result, reaching high average precision while maintaining low computational and memory consumption.
- We deploy our model on NVIDIA Jetson TX2 to validate its real-time performance. The results demonstrate that our model is remarkable efficient in practical applications and can be easily generalized to the human keypoint detection task.

2 Related Work

High Resolution Representation Tasks that require position-sensitive information rely on high-resolution representations. There are two mainstream approaches to obtaining high-resolution representations. One method is to employ a high-resolution recovery process to enhance the representation resolution. This is achieved by improving the low-resolution output obtained from a classification network[17–22] through sequentially-connected convolutions, typically upsampling. However, spatial sensitivity information has already been lost in the previous downsampling and cannot be recovered. The other way is to replace the downsampling and normal convolution layers with dilated convolutions[23–32]which will significantly increases the computational complexity and number of parameters. HRNet[10, 33] is proposed as an efficient way to maintain high-resolution representation throughout the network. HRNet consists of parallel multi-resolution branches. By repeated multi-scale fusion, HRNet can generate high-resolution representation with rich semantic.
Model Lightweighting Lightweight model are needed for practical real-time application. Separable convolutions and group convolutions, derived from classification networks[2, 3, 32, 34–39] are commonly used techniques to reduce computational redundancy. MobileNetv3 is built upon depthwise separable convolutions and introduces the inverted residual structure to construct lightweight networks. ShuffleNetv2, on the other hand, incorporates pointwise group convolutions and channel shuffling to maintain model performance. Although these models achieve relatively robust performance with fewer computations, they do not fully explore the redundancy between feature maps to further compress the model. In contrast, the Ghost module[11], proposed by Kai Han et al., can generate additional feature maps from the intrinsic features through cheap linear operations, significantly conserving computational and storage costs.
Attention Mechanisms Network lightweighting inevitably leads to the loss of spatial information. Incorporating attention mechanisms into convolutional

networks allows the model to adaptively allocate weights to different regions and highlight important features to enhance the network's representation, therefore compensate for the loss of information to some extent. Prior studies on attention mechanisms focus on enhancing the performance of neural networks by suppressing insignificant weights[13–15, 40] and lack consideration for the contribution factor of the weights, which could further suppress insignificant features. In contrast, NAM[41] uses batch normalization scaling factors to represent the importance of weights and fully exploits the contribution factor of the weights to enhance attention, which not only lowers network complexity and computational costs, but also leads to improved efficiency and model performance.

3 Approach

3.1 Parallel Multi-Branch Architecture

The model we proposed in this paper uses HRNet as the backbone to improve the performance of image processing tasks. HRNet is a high-resolution convolutional neural network capable of capturing high semantic information while maintaining high-resolution representations. It is characterized by two key features: parallel multi-resolution representations and repeated multi-resolution fusion.

Parallel Multi-Resolution Representation Starting with a high-resolution representation as the first stage, each subsequent stage includes the previous stage's resolution representation and expands a lower-resolution representation, as shown in **Figure 2**. The resolutions of all four representations are 1/4, 1/8, 1/16, and 1/32, respectively. Representations of different resolutions are connected in parallel to avoid information loss. Also feature extraction are simultaneously performed on multiple scales.

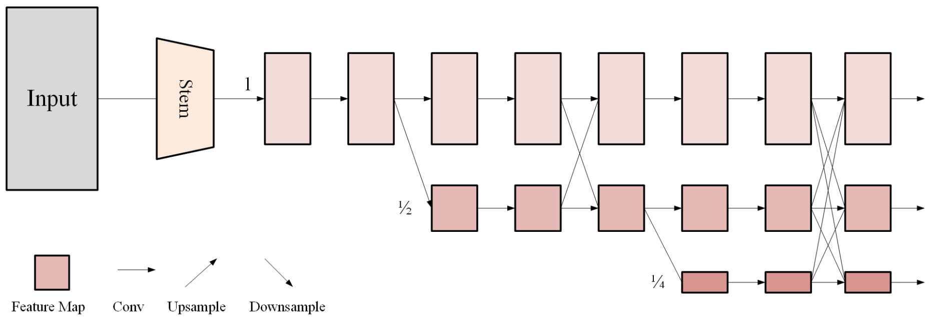


Fig. 2: Parallel Multi-Resolution Representation

Repeated Multi-Resolution Fusion The purpose of fusion is to exchange information between representations of different resolutions. Take the fusion of

2-resolution representations for example, the input consists of three representations: $\{R_r^i, r = 1, 2\}$, with r is the resolution index, and the associated output representations are $\{R_r^o, r = 1, 2\}$. Each output representation is the sum of the transformed representations of the two inputs:$R_r^o = f_{1r}(R_1^i) + f_{2r}(R_2^i)$. There will be an additional output from stage 2 to stage 3, which is the expanded lower-resolution representation: $R_3^o = f_{13}(R_1^i) + f_{23}(R_2^i)$.The choice of the transform function $f_{xr}(\cdot)$ is dependent on the input resolution index x and the output resolution index r. If $x = r$, $f_{xr}(R) = R$. If $x < r$, $f_{xr}(R)$ downsamples the input representation R through (rs) stride-2 3×3 convolutions.If $x > r$, $f_{xr}(R)$ upsamples the input representation R through the bilinear upsampling followed by a 1×1 convolution for aligning the number of channels. The functions are depicted in **Figure 3**.

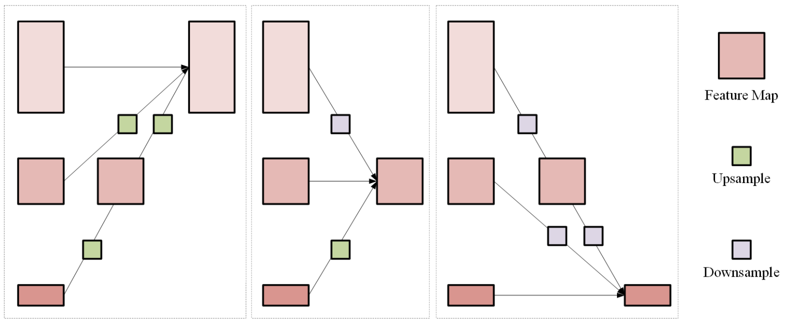


Fig. 3: Repeated Multi-Resolution Fusion

The low-resolution branch can capture the global features of the image, while the high-resolution branch can preserve more local details. HRNet performs well in various image processing tasks, especially for tasks that require high-resolution representations, such as human pose estimation, semantic segmentation, and object detection.

3.2 Model Lightweighting

Considering the limited resources of embedded devices, model lightweighting is necessary to deploy convolutional neural networks. The Ghost module can be used as a plug-and-play component to accelerate model.

Key of the Ghost module is cheap linear operation. Given the input data $X \in R^{c \times h \times w}$, the operation of an arbitrary convolutional layer for producing n feature maps can be formulated as **Equation 1**:

$$Y = X * f + b \tag{1}$$

where $*$ is the convolution operation, b is the bias term, $Y \in R^{h' \times w' \times n}$ is the output feature map, $f \in R^{c \times k \times k \times n}$ is the convolution filters, and the convolution kernel is k. Taking into account the similarity between the generated feature maps, these redundant feature maps are called ghost feature maps. Suppose that those ghost feature maps can be easily transformed from smaller-scale intrinsic feature maps, we can modify the above equation as follows, as **Equation 2 and Equation 3**:

$$Y' = X * f' \tag{2}$$

$$y_{ij} = \Phi_{i,j}\left(y'_i\right), \quad \forall i = 1, \ldots, m, \quad j = 1, \ldots, s \tag{3}$$

where $f' \in R^{c \times k \times k \times m}$ is the utilized filters, $m \leq n$,the bias term is omitted for simplicity, y'_i is the i-th intrinsic feature map in Y', and $\Phi_{i,j}$ is the j_{th} linear operation used to generate the j_{th} ghost feature map y_{ij}. The last $\Phi_{i,s}$ is the identity mapping for preserving the intrinsic feature maps. We can obtain $m = n \cdot s$ feature maps as shown in **Figure 4.**

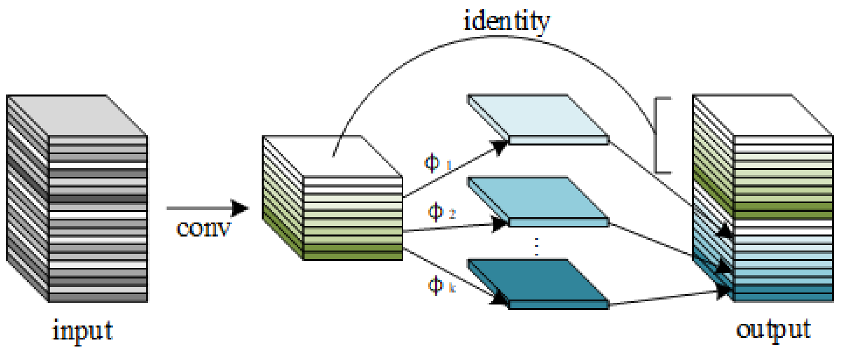


Fig. 4: Cheap Operation

3.3 Adaptive Attention Mechanism

In order to improve the accuracy and robustness of the model without increasing the complexity of the network, we introduce the adaptive attention mechanism NAM. NAM consists of two sub-modules: the channel attention and the spatial attention sub-module.

For the channel attention sub-module, NAM uses a scaling factor from BN to reflect the importance of channels, as shown in **Equation 4** .

$$B_{out} = BN(B_{in}) = \gamma \frac{B_{in} - \mu B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{4}$$

where μB and σB are the mean and standard deviation of the mini-batch B, respectively; γ and β are trainable affine transformation parameters for scaling and translation operations, so that the model can adaptively adjust the activated range and central location. The process of weighting channels is shown in **Figure 5**, where F_1 is the input feature map, γ_i is the scaling factor of each channel, and its weight is $W_i = \gamma_i / \sum_{j=0}^{\gamma_j}$.

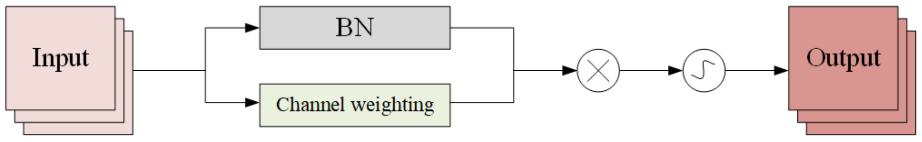


Fig. 5: Channel Attention Submodule

The weighting process of spatial attention submodule is similar to channel weighting, as illustrated in **Figure 6**, where the output is denoted as M_s, λ is the scaling factor, and the weights are $W_i = \lambda_i / \sum_{j=0}^{\lambda_j}$.

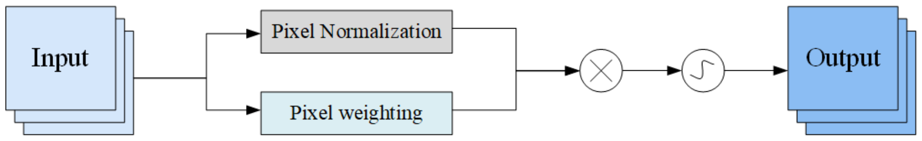


Fig. 6: Spacial Attention Submodule

In addition , NAM also introduces $L1$ regularization as a weight sparsity penalty in the loss function,as shown in **Equation 5**, where x represents the input ,y represents the output , W represents the network weights , $l(\cdot)$ is the loss function; $g(\cdot)$ is the $L1$ norm penalty function and p is uEsed to balance $g(\gamma)$ and $g(\lambda)$ penalty item. By introducing sparsity constraints into model weights, the model is prompted to focus on more important features while reducing unnecessary calculations.

$$LOSS = \sum_{(x,y)} l(f(x,W),y) + p\sum g(\gamma) + p\sum g(\lambda) \tag{5}$$

3.4 GoatPose

The GoatPose model proposed in this paper is a lightweight high-resolution network. Using HRNet as the backbone, GoatPose redesigns the convolution unit and introduces cheap operation into the conventional convolution module to reduce the amount of parameters. In order to compensate for the information loss and further strengthen feature extraction, GoatPose incorporates the NAM attention mechanism into its architecture. Goat symbolizes that the model in this paper runs as quickly and lightly as a goat. Just as a goat's two horns stand out, certain channels or features in the network receive more attention due to their higher importance in the task.

Design of Convolution Block We redesign the normal convolution block by incorporating cheap linear operation, and name this modified block LiteConv, as shown in **Figure 7**. The purpose is to obtain rich feature maps with limited computational resources. For each LiteConv block, the input X first passes through a normal convolution, followed by BatchNorm and ReLU to upsample the input feature maps. The resulting feature maps are denoted as $X1$. Then, we conduct cheap linear operation on the intrinsic feature map$X1$to generate Ghost feature maps $X2$, including a sequence of a depthwise separable convolution, BatchNorm and ReLU, and the output is concatenated with the previous output $X1$. Considering that the normal convolution block is the most frequently-used part, substituting it with LiteConv block not only reduces the computational burden but also improves the overall efficiency of the model by reducing the number of parameters.

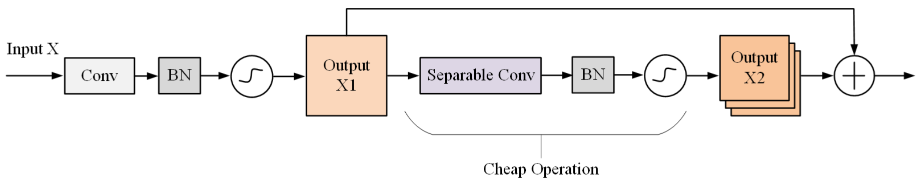


Fig. 7: LiteConv Block

Compensate for the Information Loss Considering that replacing convolutions with linear operations for feature map generation can inevitably lead to information loss, we introduce the NAM attention mechanism to compensate for this loss. The NAM module is inserted after the stack of Basic Blocks to timely augment the feature representation.

Lightweight High-Resolution Network GhostNet consists of four stages. Starting from a high-resolution convolutional stream $W \times H \times C$, the first stage includes 4 bottlenecks for extracting image features and downsampling. Then a 3×3 convolution is applied to generate an additional haif-resolution stream $W/2 \times H/2 \times 2C$. The two convolutional streams output to the next stage in parallel. The second, third, and fourth stages consist of 1, 4, and 3 Stage Modules, respectively.

Each Stage Module contains parallel multi-resolution convolutional streams and implements fusion of different resolution streams. It consists of Basic block, channel attention module NAM-C, and fusion unit. The convolutional streams of different resolutions first pass through four Basic Blocks and then go through the NAM-C module. Finally, they are fused with other convolutional streams through the fusion unit.

The basic block consists of four residual units, each containing two 3×3 convolutions followed by BN and ReLU. In the NAM-C module, the input X is first normalized using a BN layer. Then we compute the weights for channel-wise normalization. The weighted feature maps are obtained by element-wise multiplication of the weights with the input. Finally, the weighted feature maps are scaled using the sigmoid function to obtain the adjusted feature maps through the channel attention mechanism. The fusion unit connects the outputs of different resolutions in a fully connected manner, as illustrated in the previous example. Transition modules are used between stages to expand the convolutional streams, adding an additional parallel convolutional stream with halve resolution using a 3×3 convolution. This stream is then concatenated with the convolutional streams from the previous stage and serves as the input for the next stage.

The GhostPose model is a lightweight convolutional network suitable for deployment on small embedded devices for human pose estimation, as shown in **Figure 8**. It significantly reduces computational complexity and model parameter count while maintaining high accuracy.

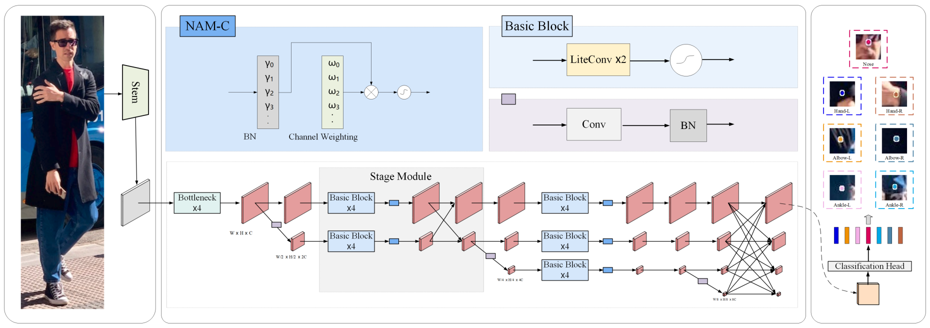


Fig. 8: GoatPose Architecture

4 Experiments

4.1 Dataset

Microsoft COCO Our model is evaluated on the most popular **MSCOCO** dataset for human pose estimation, which contains an extensive collection of approximately 200,000 images and 250,000 person examples. For training, we use the train2017 dataset consisting of 57,000 images and 150,000 person instances, and the val2017 dataset containing 5,000 images is used for the evaluation process.

Evaluation Metric In the MS's COCO2017 dataset, the standard evaluation metric is based on Object Keypoint Similarity (OKS),as shown in **Equation 6**:

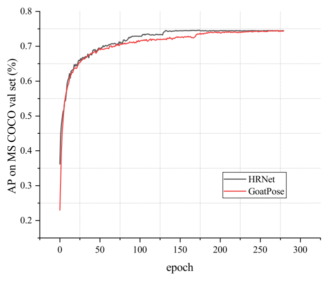
$$\mathrm{OKS} = \frac{\sum_i \exp\left(-\frac{d_i^2}{2s^2k_i^2}\right)\delta\left(v_i > 0\right)}{\sum_{\mathrm{i}} \delta\left(v_i > 0\right)} \tag{6}$$

where d_i is the Euclidean distance between a detected keypoint and its corresponding ground truth position, s is the object scale, k_i is a per-keypoint constant that controls falloff, and v_i denotes the visibility flag of keypoint i.

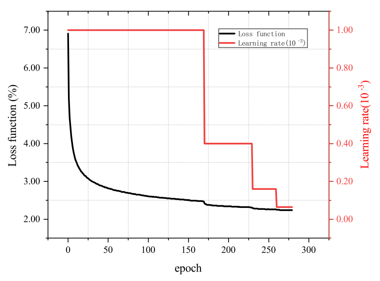
We report standard average precision and recall scores, as shown in **Figure 9a**: AP50 (AP at OKS = 0.50), AP75, AP (the mean of AP scores at OKS = 0.50, 0.55, . . . , 0.90, 0.95), APM for medium objects, APL for large objects, and AR (the mean of recalls at OKS = 0.50, 0.55, . . . , 0.90, 0.95).

4.2 Setting

The network is trained on NVIDIA GeForce RTX 308012G, and the installed CUDA Version is Cuda12.0. We train the model for a total of 280 epochs. We adopt Adam optimizer with an initial learning rate of 0.001 and a decay rate of 0.4. The base learning rate dropped to 0.0004 and 0.00016 at the 170th and 230th and 260th epoch respectively,as shown in **Figure 9b**.



(a) Accuracy



(b) Loss

Fig. 9: Accuracy and Loss

4.3 Results

Comparisons on COCO dataset By adjusting the resolution and number of channels of the input image, we construct four versions of GoatPose, each with the same hyperparameter configuration. Among them, B means that the model input has 32 channels, X represents the model input has 48 channels; L means the size of the input is 256×192, and H means the size of the input image is 384×288. In the following context We mainly analyze the performance improvement of GoatPose-BL.

Table 1 reports the comparison of AP and AR score between our networks and other state-of-the-art methods. Compared to models based on the HRNet-W32 backbone, GoatPose-BL achieves similar average precision to HRNet-W32L and TokenPose-B, and outperforms TransPose-H-S by 1.5%. Meanwhile, GoatPose-BL shows significant improvements in AR score over all aforementioned models, with increases of 3% and 2.9% compared to HRNet-W32 and TokenPose-B, respectively. **Table 2** presents the comparison of complexity between our networks and other state-of-the-art methods. GoatPose-BL has fewer parameters compared to the majority of models, meanwhile, its execution speed is significantly faster than all other models. GoatPose-BL has only 15.51M parameters, which is nearly half of the parametes of HRNet-W32L. Additionally, its computational speed is 3.48 GFLOPs, which is approximately twice as fast as HRNet-W32L.Compared to the model TransPose-H-S with the least parameters, GoatPose has slightly more parameters but achieves a four-fold improvement in computation speed.

Table 1: Comparison on Accuracy.

Model	Backbone	Input Size	Feature Size	AP	AP50	AP75	AR
Simple Baseline	ResNet-152	256 × 192	1/32	72	89.3	79.8	77.8
TokenPose-B	HRNet-W32	256 × 192	1/4	74.7	89.8	81.4	80.0
TokenPose-L/D6	HRNet-W48	256 × 192	1/4	75.4	90.0	81.8	80.4
TokenPose-L/D24	HRNet-W48	256 × 192	1/4	75.8	90.3	82.5	80.9
TransPose-H-S	HRNet-W32	256 × 192	1/4	73.4	91.6	81.1	-
TransPose-H-A4	HRNet-W48	256 × 192	1/4	74.7	91.9	82.2	-
TransPose-H-A6	HRNet-W48	256 × 192	1/4	75	92.2	82.3	80.8
HRFormer-BL	HRFormer-B	256 × 192	1/4	75.6	90.8	82.8	80.8
HRFormer-BH	HRFormer-B	384 × 228	1/4	77.2	91	83.6	82
ViTPose-B	ViT-B	256 × 192	1/16	75.8	90.7	-	81.1
ViTPose-L	ViT-L	256 × 192	1/16	78.3	91.4	-	83.5
HRNet-W32L	HRNet-W32	256 × 192	1/4	74.4	90.5	81.9	79.8
HRNet-W32H	HRNet-W32	384 × 288	1/4	75.8	90.6	82.7	81
HRNet-W48L	HRNet-W48	256 × 192	1/4	75.1	90.6	82.2	80.4
HRNet-W48H	HRNet-W48	384 × 288	1/4	76.3	90.8	82.9	81.2
GoatPose-BL	HRNet-W32	256 × 192	1/4	74.5	92.6	82.5	82.3
GoatPose-BH	HRNet-W32	384 × 288	1/4	75.6	92.7	82.3	83.5
GoatPose-XL	HRNet-W48	256 × 192	1/4	76.1	93.5	83.5	83.8
GoatPose-XH	HRNet-W48	384 × 1288	1/4	76.5	93.7	83.6	83.9

Ablation Study We empirically study how cheap operation and attention mechanism influence the performance, as shown in **Table 3**. We constructed NAM-HRNet by incorporating the NAM attention mechanism onto the HRNet architecture.On COCO val, NAM-HRNet demonstrates a significant improvement in AP compared to HRNet, with an increase of 1.5%. The improvement

Table 2: Comparison on Complexity.

Model	Params(M)	GFLOPs	Speed(fps)	AP	AR
Simple Baseline	60	15.7	123	72	77.8
TokenPose-B	14	5.7	-	74.7	80
TokenPose-L/D6	21	9.1	-	75.4	80.4
TokenPose-L/D24	28	11	89	75.8	80.9
TransPose-H-S	8	10.2	54	73.4	-
TransPose-H-A4	17	17.5	49	74.4	—
TransPose-H-A6	18	21.8	46	75	80.8
HRFormer-BL	43	12.2	23	75.6	80.8
HRFormer-BH	43	26.8	12	77.2	82
ViTPose-B	86	17.1	140	75.8	81.1
ViTPose-L	307	70.0+	61	78.3	83.5
HRNet-W32L	29	7.1	136	74.4	79.8
HRNet-W32H	29	16	64	75.8	81
HRNet-W48L	64	14.6	96	75.1	80.4
HRNet-w48H	64	32.9	46	76.3	81.2
GoatPose-BL(ours)	**15.51**	**3.48**	**277**	**74.5**	**82.3**
GoatPose-BH(ours)	15.51	7.84	131	75.6	83.5
GoatPose-XL(ours)	34.76	7.81	180	76.1	83.8
GoatPose-XH(ours)	34.76	17.53	86	76.5	83.9

confirms that the weight allocation of the attention mechanism can effectively enhance model performance. Compared to NAM-HRNet , GoatPose-BL achieves an AP score of 74.5 while reducing both computational complexity and parameters by half, which validates the efficiency and effectiveness of cheap operation.

Table 3: Ablation about Cheap Operation and Attention mechanism.

Model	Params	GFLOPs	AP
HRNet	29	7.1	74.4
NAM-HRNet(ours)	35	7.8	76.1 **(up 1.5)**
GoatPose(ours)	15.51	3.48	74.5

4.4 Deployment

In order to verify the real-time performance of our model, we deploy GoatPose on NVIDIA Jetson TX2.

NVIDIA Jetson TX2 We picked NVIDIA TX2 platform as the testing platform for our model. The NVIDIA Jetson series processors are frequently used to support mainstream CNN models and seamless integration with CUDA. Additionally, they are compatible with NVIDIA's TensorRT, an accelerated inference framework that facilitates fast and user-friendly inference methods.

Model Performance After preliminary verification, we convert and deploy our model on Jetson TX2 platform. We first install JetPack on the TX2 and configure the necessary environment. Then we debug the code on a server, train the model, and conduct initial tests. Afterwards, the trained model in the .pth format is converted to a compatible format trt with the TensorRT acceleration module. We use TensorRT to accelerate the converted model and perform inference.Our testing result shown in **Tabel 4** demonstrates that GoatPose can achieve excellent performance with a high speed even under limited hardware resources.

Table 4: Application on Jetson TX2.

Heading level	Example	Font size and style	speed(TX2)	AP(%)
HRNet-W32L	29	7.1	23	74.4
GoatPose-BL(ours)	15.51	3.48	**51**	74.5

5 Conclusion

By incorporating cheap linear operations, we novelly propose a a lightweight deep convolutional network GoatPose, which can reduce the amount of computation and parameters by half while maintaining the same or even slightly higher accuracy compared with the backbone HRNet. Specifically, the lightweight modules are combined with NAM attention machanism, greatly improve the effectiveness of GPU computing. Experiments on COCO dataset shows that our model can boost the sample efficiency of the baseline HRNet, supported by significantly improved performance. The successful deployment on NVIDIA Jetson TX2 further demonstrates the superiority and generalizability of our model , paving the way to real-time human pose estimation for edge applications.

References

1. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 6848–6856 (2017)
2. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017), http://arxiv.org/abs/1704.04861, cite arxiv:1704.04861
3. Howard, A., Pang, R., Adam, H., Le, Q.V., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y.: Searching for mobilenetv3. In: ICCV. pp. 1314–1324. IEEE (2019)
4. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks (2017)

5. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming (2017)
6. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding (2016)
7. Lin, J., Rao, Y., Lu, J., Zhou, J.: Runtime neural pruning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 2178–2188. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
8. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks (2016)
9. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. p. 1135–1143. NIPS'15, MIT Press, Cambridge, MA, USA (2015)
10. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation (2019)
11. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: Ghostnet: More features from cheap operations (2020)
12. Han, K., Wang, Y., Xu, C., Guo, J., Xu, C., Wu, E., Tian, Q.: GhostNets on heterogeneous devices via cheap operations. International Journal of Computer Vision **130**(4), 1050–1069 (mar 2022). https://doi.org/10.1007/s11263-022-01575-y, https://doi.org/10.10072Fs11263-022-01575-y
13. Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks (2019)
14. Park, J., Woo, S., Lee, J.Y., Kweon, I.S.: Bam: Bottleneck attention module (2018)
15. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: Convolutional block attention module (2018)
16. 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 (2015)
17. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation (2016)
18. Badrinarayanan, V., Handa, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling (2015)
19. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation (2015)
20. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015)
21. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking (2018)
22. Peng, X., Feris, R.S., Wang, X., Metaxas, D.N.: A recurrent encoder-decoder network for sequential face alignment (2016)
23. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions (2016)
24. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs (2016)
25. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs (2017)
26. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation (2017)

27. Chen, L.C., Collins, M.D., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., Shlens, J.: Searching for efficient multi-scale architectures for dense image prediction (2018)
28. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation (2019)
29. Yang, T.J., Collins, M.D., Zhu, Y., Hwang, J.J., Liu, T., Zhang, X., Sze, V., Papandreou, G., Chen, L.C.: Deeperlab: Single-shot image parser (2019)
30. Cheng, B., Collins, M.D., Zhu, Y., Liu, T., Huang, T.S., Adam, H., Chen, L.C.: Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation (2020)
31. Cheng, B., Collins, M.D., Zhu, Y., Liu, T., Huang, T.S., Adam, H., Chen, L.C.: Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation (2020)
32. Niu, Y., Wang, A., Wang, X., Wu, S.: Convpose: A modern pure convnet for human pose estimation. Neurocomputing **544**, 126301 (05 2023). https://doi.org/10.1016/j.neucom.2023.126301
33. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., Xiao, B.: Deep high-resolution representation learning for visual recognition (2020)
34. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks (2019)
35. Chollet, F.: Xception: Deep learning with depthwise separable convolutions (2017)
36. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017)
37. Sun, K., Li, M., Liu, D., Wang, J.: Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks (2018)
38. Tan, M., Le, Q.V.: Mixconv: Mixed depthwise convolutional kernels (2019)
39. Neff, C., Sheth, A., Furgurson, S., Tabkhi, H.: Efficienthrnet: Efficient scaling for lightweight high-resolution multi-person pose estimation (07 2020)
40. Liu, Z., Wang, L., Wu, W., Qian, C., Lu, T.: Tam: Temporal adaptive module for video recognition (2021)
41. Liu, Y., Shao, Z., Teng, Y., Hoffmann, N.: Nam: Normalization-based attention module (2021)