Computational Visual Media
https://doi.org/10.1007/s41095-021-0214-z Vol. 7, No. 3, September 2021, 335–347
Research Article

EfficientPose: Efficient human pose estimation with neural architecture search

Wenqiang Zhang[1,*], Jiemin Fang[2,1,*], Xinggang Wang[1] (✉), and Wenyu Liu[1]

Abstract Human pose estimation from image and video is a key task in many multimedia applications. Previous methods achieve great performance but rarely take efficiency into consideration, which makes it difficult to implement the networks on lightweight devices. Nowadays, real-time multimedia applications call for more efficient models for better interaction. Moreover, most deep neural networks for pose estimation directly reuse networks designed for image classification as the backbone, which are not optimized for the pose estimation task. In this paper, we propose an efficient framework for human pose estimation with two parts, an efficient backbone and an efficient head. By implementing a differentiable neural architecture search method, we customize the backbone network design for pose estimation, and reduce computational cost with negligible accuracy degradation. For the efficient head, we slim the transposed convolutions and propose a spatial information correction module to promote the performance of the final prediction. In experiments, we evaluate our networks on the MPII and COCO datasets. Our smallest model requires only 0.65 GFLOPs with 88.1% PCKh@0.5 on MPII and our large model needs only 2 GFLOPs while its accuracy is competitive with the state-of-the-art large model, HRNet, which takes 9.5 GFLOPs.

Keywords pose estimation; neural architecture search; efficient deep learning

1 Introduction

Human pose estimation has a wide range of multimedia applications in the real world, e.g., in virtual reality, game interaction, and assisted living. Human pose estimation aims to determine the locations of human keypoints or parts in images. Traditional methods [1, 2] are based on a probabilistic graphical model or pictorial structures. Deep convolutional neural networks boosted the development of this field by directly regressing keypoint positions [3] or predicting heatmap locations [4], solving this problem in an end-to-end manner. Later top–down methods [5, 6] achieved high accuracies on both MPII [7] and COCO [8] benchmarks. However, the backbone networks in previous methods [5, 9, 10] often directly reuse networks designed for image classification, e.g., ResNet [11], which results in sub-optimality for human pose estimation tasks. Moreover, existing human pose estimation methods over-pursue accuracy but ignore the efficiency of the model, making it difficult to deploy the model on resource-constrained computing devices commonly used in real-life scenarios. Recent multimedia applications require more efficient human pose estimation to bring a better interactive experience for users. However, current pose estimation algorithms cannot meet this requirement.

In recent years, the emergence of neural architecture search (NAS) methods has greatly accelerated the development of neural network design. Some pioneering works [13, 14] required a huge search cost to obtain an architecture with high accuracy on image classification tasks. Later one-shot and differentiable NAS methods [15, 16] greatly decreased the search cost while keeping the high performance of the architectures found. NAS has also been applied

* Wenqian Zhang and Jiemin Fang contributed equally to this work.
1 School of EIC, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: W. Zhang, wq_zhang@hust.edu.cn; X. Wang, xgwang@hust.edu.cn (✉); W. Liu, liuwy@hust.edu.cn.
2 Institute of Artificial Intelligence, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: jaminfong@hust.edu.cn.
Manuscript received: 2020-12-11; accepted: 2021-02-16

to cut down the required number of FLOPs or latency of the architecture while guaranteeing high accuracy [17, 18]. Furthermore, some methods directly search for customized architecture designs or performance improvements in, e.g., semantic segmentation [19, 20] and object detection [21, 22]. However, searching for backbone networks for segmentation, detection, and pose estimation is very computationally expensive. In previous pose estimation methods, customized and automatic backbone design is rarely explored. To optimize networks, searching for a backbone is important as it forms a large part of the whole network, and performs feature extraction.

To directly search for a backbone for pose estimation, we propose to design an efficient human pose estimation network search framework, which we call EfficientPose. As shown in Fig. 1, our efficient network includes two main parts, an efficient backbone and an efficient head. To tackle the backbone performance bottleneck in terms of both accuracy and efficiency, the differentiable NAS method [16] is designed to lighten the computational cost of the backbone network and to adapt the backbone architecture to pose estimation tasks. We further design an efficient pose estimation head which enables not only fast inferencing but also fast architecture search. In the head network, the transposed convolutions are slimmed according to the width of the backbone. A *spatial information correction* (SIC) module is proposed to promote the quality of the feature maps used for generating the heatmaps. Overall, we promote the efficiency of the whole framework for pose estimation. The architectures generated by our framework possess high performance with low numbers of required FLOPs and latency, as shown in Fig. 2.

Our contributions can be summarized as follows.

- An efficient pose estimation network search framework. The differentiable NAS method adjusts and lightens the backbone network automatically. The head network is redesigned by the proposed slim transposed convolutions and spatial information correction module to give a better trade-off between computational cost and accuracy.
- An extremely efficient pose estimation network that requires only 0.65 GFLOPs yet has high performance (88.1% PCKh@0.5). Our large model takes only 2 GFLOPs while its accuracy is competitive with the state-of-the-art large model, HRNet [6] which takes 9.5 GFLOPs. Furthermore, the good generalizability of the proposed EfficientPose networks has been confirmed on the COCO dataset.
- An extensive analysis and study on the design principle of our efficient framework, providing heuristic conclusions for future network design for human pose estimation.

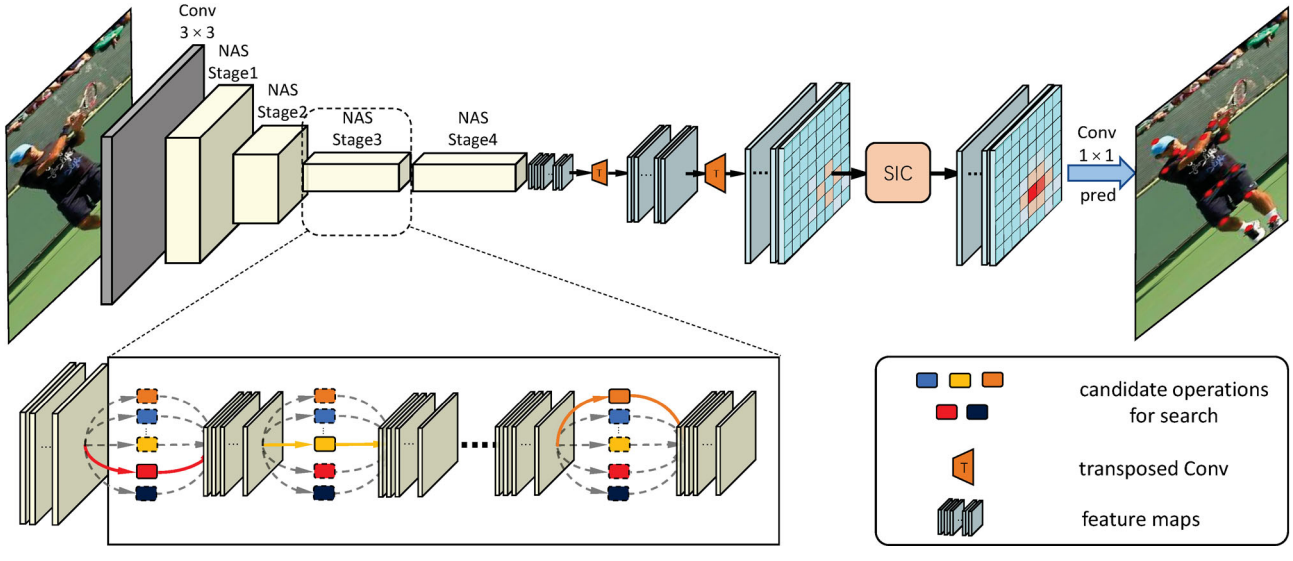

Fig. 1 EfficientPose framework. For the backbone, we use the NAS method to obtain more efficient and precise networks. The desired network is finally derived from the super network. For the head part, we slim the transposed convolution and use a SIC module to correct the spatial information in the feature maps after the transposed convolutions. The input "Conv3 × 3" denotes a plain 3 × 3 convolution followed by a 3 × 3 separable depthwise convolution [12].

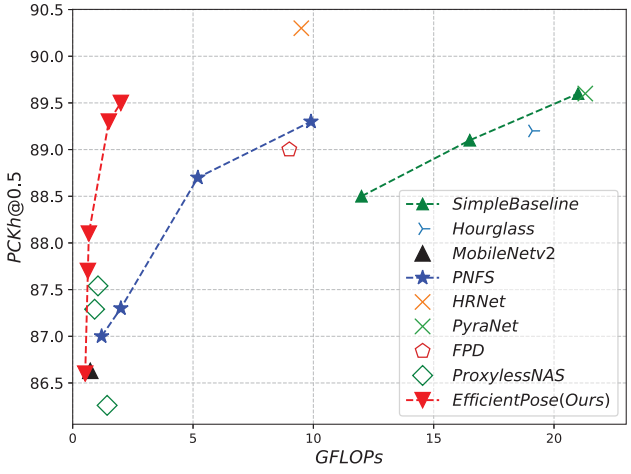

Fig. 2 FLOPs needed by different networks for the MPII validation dataset.

2 Related work

2.1 Human pose estimation

Previous methods of human pose estimation have achieved great success. DeepPose [3] first applied deep learning to pose estimation tasks, regarding pose estimation as a regression problem. This convolutional neural network outputs the keypoints positions directly. Most work in recent years predicts the locations of heat maps, which allows the network to save more spatial information. Hourglass [4] stacks repeating U-shaped blocks with skip connections to promote overall performance. PyraNet [23] uses a pyramid residual module to obtain multi-scale information. CPN [9] adopts a GlobalNet to roughly localize keypoints and a RefineNet to explicitly handle hard keypoints. SimpleBaseline [5] aims at constructing a simplified network and applies transposed convolutions to get high-resolution heatmaps. HRNet [6] achieves state-of-the-art results on COCO [8] by maintaining high-resolution representations, but the multi-branch framework leads to a high computational cost and is not friendly for inferencing on embedded devices.

Though previous methods have achieved high accuracies in human pose estimation tasks, most suffer from high computational cost and high latency. The efficiency of the network is rarely considered, which makes them difficult to apply in real-word scenarios. Bulat and Tzimiropoulos [24] improved performance of the binarized model to get a better trade-off between model size and accuracy. DU-Net [25] applies quantification to reduce model size. FPD [26] uses a strong teacher network to supervise small student networks to improve performance. We aim to construct a lightweight framework for pose estimation which achieves high accuracy with a low computational cost.

2.2 Spatial information correction

When processing pixel-level tasks, neural networks usually downsample the input image first, and then upsample the encoded features. Previous works [10, 27] use interpolation to handle this task, or use learnable transposed convolutions to improve performance [5]. These approaches expand the receptive field and make the model more efficient, but at the same time cause the loss of feature information. Transposed convolutions often cause a checkerboard pattern of artifacts [28]. To solve this problem, different upsampling layer designs [28–30] have been proposed. Sugawara et al. [31] addressed the issue that CNNs must have a nonperiodic steady-state value in the unit step response to avoid checkerboard artifacts, providing insights for our design.

2.3 Neural architecture search

Recent NAS methods greatly improve the performance of neural networks. Early NAS works use reinforcement learning [13, 32] or an evolutionary algorithm [14] to search for architectures and achieve superior results to networks designed manually. Later one-shot [15, 33] or differentiable [16–18, 34, 35] NAS methods search for high-performance architectures with low computational cost. To further reduce the search cost, some methods search for a cell structure and then stack it to build the final architecture [16, 35]. NAS is also applied to improve model efficiency [17, 18, 32, 36] by optimizing the required number of FLOPs, latency, etc. NAS methods have already been applied to other tasks, e.g., semantic segmentation [19, 20] and object detection [21, 22]. In this paper, we implement a differentiable NAS to backbone network design for human pose estimation. The resulting networks achieve a better trade-off between accuracy and efficiency than other state-of-the-art methods [5, 6, 37].

3 Method

In this section, we first introduce the method of

differentiable neural architecture search, used to design the backbone network automatically targeted at pose estimation. Secondly, we explain how the search space for the backbone network is designed and how we optimize both accuracy and latency. Finally, we illustrate the design principles of our redesigned lightweight head. The whole framework is shown in Fig. 1.

3.1 Differentiable neural architecture search

We use differentiable NAS [16–18] to customize the backbone network design for the pose estimation task. We formulate the NAS problem as a nested optimization problem:

$$\min_{a\in\mathcal{S}} \min_{w_a} \mathcal{L}(a, w_a) \tag{1}$$

where $\mathcal{S}$ represents the search space and w_a denotes the operating weights of architecture a. We search for the architecture a by minimizing the loss $\mathcal{L}(a, w_a)$

In the differentiable NAS method, the search space $\mathcal{S}$ is relaxed into a continuous representation. Specifically, in every layer of the architecture, we compute the probability of each candidate operation as

$$p_o^\ell = \frac{\exp(\alpha_o^\ell)}{\sum_{o'\in\mathcal{O}} \exp(\alpha_{o'}^\ell)} \tag{2}$$

where $\mathcal{O}$ denotes the set of candidate operations and α_o^ℓ denotes the architectural parameter of operation o in layer ℓ. The output of each layer is computed as a weighted sum of output tensors from candidate operations:

$$x_{\ell+1} = \sum_{o\in\mathcal{O}} p_o^\ell \cdot o(x_\ell) \tag{3}$$

where x_ℓ denotes the input tensor of layer ℓ.

During the search process, the operating weights and architectural parameters are optimized alternately by stochastic gradient descent to minimize the loss $\mathcal{L}$. The final architecture is derived based on the distribution of architectural parameters.

3.2 Efficient backbone

Most previous methods [5, 9, 10, 27] use image classification networks as the backbone for pose estimation, e.g., ResNet [11], MobileNetV2 [38]. To narrow the gap between classification and pose estimation tasks, we redesign the backbone network using the NAS method. We first give details of the search space, and then we describe how we optimize both accuracy and latency of the model.

3.2.1 Search space

We construct our search space based on the MobileNetV2 [38] network, which provides great performance at low computational cost and is commonly used to design the search space in NAS methods [17, 18]. We stack inverted residual blocks (i.e., MBConvs) to build the backbone network and allow MBConvs to have various settings. Specifically, kernel sizes include {3, 5, 7} and expansion ratios include {3, 6}. Skip connections are used in candidate operations when determining the depth of the network. The dropping-path training strategy [15, 17, 39] is used to decrease coupling between different sub-architectures in the search space.

Unlike previous methods [5, 6], we only perform 4 down-sampling operations in the backbone network (5 are used in most other methods). We consider that in a lightweight pose estimation network, 5× down-sampled feature maps contribute little to the final prediction and may lose much information. The higher-resolution (4×) feature maps are used for up-sampling. We give details of the backbone network in Table 1.

3.3 Cost optimization

Cost optimization of a neural network is of critical importance and can be measured by different benchmarks such as FLOPs or latency. To obtain a high-performance network with low cost, we integrate cost optimization into the search formulation.

Table 1 Search space for the backbone. The first "Conv3×3" is a plain convolution with a 3 × 3 kernel. "SepDepth3 × 3" is a separable depthwise convolution [12] with kernel size 3 × 3. "TransConv" is the transposed convolution. s2 denotes a stride of 2. EfficientPose-A is found using the small setting for the search space, while EfficientPose-B and -C are found using the large setting

Stage	Output size	Layer	
		Small	Large
Input	256 × 256	—	
Conv3 × 3	128 × 128	[32, s2]	
SepDepth3 × 3	128 × 128	[16, s1]	[24, s1]
NAS Stage1	64 × 64	[24, s2] [24, s1] × 3	[32, s2] [32, s1] × 5
NAS Stage2	32 × 32	[32, s2] [32, s1] × 5	[64, s2] [64, s1] × 7
NAS Stage3	16 × 16	[64, s2] [64, s1] × 9	[96, s2] [96, s1] × 9
NAS Stage4	16 × 16	[96, s1] × 8	[160, s2] × 10
TransConv	32 × 32	[64, s2]	
TransConv	64 × 64	[32, s2]	

Following most differentiable NAS methods [17, 18, 39], we build a lookup table to predict the cost of the architecture during search. The cost of one layer is computed as

$$\text{cost}_\ell = \sum_{o\in\mathcal{O}} p_o^\ell \text{cost}_o^\ell \tag{4}$$

where cost_o^ℓ is the cost of operation o in layer ℓ and p_o^ℓ denotes the corresponding probability computed by architectural parameters. The latency of the whole network can be computed as

$$\text{cost} = \sum_i \text{cost}_i \tag{5}$$

We add cost regularization to the loss function for multi-objective optimization. The loss function during search is defined as

$$\begin{aligned} \mathcal{L}(a, w_a) &= \mathcal{L}_{\text{MSE}} + \lambda \log_\tau \text{cost} \\ \mathcal{L}_{\text{MSE}} &= \frac{1}{K}\sum_{k=1}^{K} \|m_k - \hat{m}_k\|_2^2 \end{aligned} \tag{6}$$

where $\hat{m}_k$ is the predicted heatmap of the kth joint while m_k is the ground truth; λ and τ are hyperparameters to balance MSE loss and latency regularization.

3.4 Efficient head

The head of the pose estimation network aims to generate high-resolution heatmaps. To obtain a more efficient network, we redesign the head of the network. We first propose slim transposed convolutions which produce high-quality heatmaps with greatly decreased computational cost. We also propose a SIC module which makes the spatial information of the high-resolution representation more reliable. The SIC module improves the prediction performance at negligible computational cost.

3.4.1 Slim transposed convolutions

Unlike previous methods [5, 6], the backbone of our network outputs feature maps with a small number of channels. Accordingly, we cut down the channels of the transposed convolutions. Experiments show the effectiveness of our slim transposed convolutions. Moreover, we explored more efficient upsampling convolutions in experiments, e.g., we used separable depthwise convolution [12] to perform upsampling, and achieved good performance as well.

3.4.2 Spatial information correction

As Fig. 9 later shows, after the transposed convolutions, the feature maps present a checkerboard pattern of artifacts [28], which is caused by the uneven overlap of the transposed convolutions. To improve the quality of feature maps for heatmap generation, we add a SIC module, a 3×3 depthwise convolution, after the transposed convolutions. The SIC module only encodes features in the spatial dimension, which efficiently corrects features after the transposed convolution. As shown in Fig. 3, imbalance of learned weights can easily lead to checkerboard artifacts, even if the kernel sizes and strides of the transpose convolutions are carefully chosen. The depth-wise kernel has a good smoothing effect on the boundary without destroying the learned features.

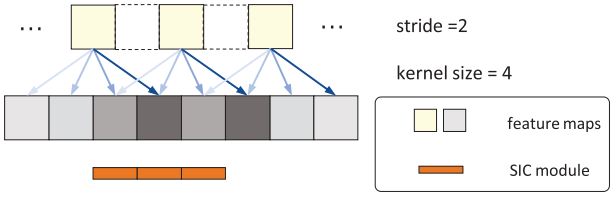

Fig. 3 Effects of the SIC module. Arrows represent convolution operations.

The SIC module almost eliminates the checkerboard artifact pattern and remarkably improves prediction performance at negligible computational cost—see later.

4 Experiments

In this section, we first describe experiments on the MPII [7] dataset. We give implementation details and compare the EfficientPose networks with other state-of-the-art (SOTA) methods and different backbone networks. Then the generalizability of EfficientPose networks is demonstrated on the COCO [8] dataset. We further specialize EfficientPose network design on different model cost benchmarks. Finally, ablation studies are carried out to show the effectiveness and efficiency of different modules in our framework.

4.1 Experiments on MPII

4.1.1 Implementation details

The MPII [7] dataset contains approximately 25k images with about 40k people. Following standard training settings [5, 6, 23], all input images are cropped to 256×256 for fair comparisons. For the backbone architecture search, we randomly split the training data into two parts, 80% for operating weight training and 20% as the validation set to update architectural parameters. The original validation set

is not used in the search process. We adopt the same data augmentation strategy as SimpleBaseline [5].

Before the search process, we build a lookup table for the latency of each operation. The latency is measured on a single GeForce RTX 2080 Ti GPU with a batch size of 32. For the backbone architecture search, we first train the operating weights for 60 epochs which takes 6.5 hours on two 2080 Ti GPUs. Then we start the joint optimization by alternately training operating weights and architectural parameters in each epoch. To update operating weights, we use the SGD optimizer with 0.9 momentum and 10^{-4} weight decay. The initial learning rate is set to 0.05 and gradually decays to 0 following a cosine annealing schedule. For architectural parameter optimization, we use the Adam optimizer with a fixed learning rate of 3×10^{-4}. The batch size is set to 32. The joint optimization process takes 150 epochs, 19 hours on two 2080 Ti GPUs. The whole search process only takes 25.5 hours on two GPUs, for 51 GPU hours in total.

We train the derived network for 200 epochs using the Adam optimizer with an initial learning rate of 10^{-3} and a batch size of 32. The learning rate decays by 0.1 at 150 and 170 epochs respectively. The fast neural network adaptation method (FNA) [22] is used to initialize both the super network in the search process and the derived network for efficient training.

4.1.2 Comparisons with SOTA pose estimation networks

Results on the MPII validation dataset are shown in Table 2. We search for three networks with different scales and compare them with the SOTA methods. Our EfficientPose networks achieve competitive and even higher performance with far fewer FLOPs. Compared with CPMs [27] and DLCM [40], our EfficientPose-A takes only 0.7 GFLOPs with 13.73 ms GPU latency while obtaining asimilar accuracy. The cost of EfficientPose-B is only 1.5 GFLOPs with 23.95 ms latency, while its performance surpasses other popular methods, e.g., Hourglass [4], SimpleBaseline(ResNet101) [5] (45.31 ms), and the knowledge distillation based method FPD [26]. Compared to PNFS [41], which searches for a cell-based fabric architecture in the head part, EfficientPose-A and -B achieve higher or similar PCKh@0.5 with 1/2.9 and 1/6.6 FLOPs. The performance of our large model EfficientPose-C is

Table 2 Comparisons with SOTA methods on the MPII validation set

Method	Params	GFLOPs	PCKh@0.5
CPMs [27]	31.0M	175.0	88.0
DLCM [40]	15.5M	33.6	87.5
SimpleBaseline-R50 [5]	34.0M	12.0	88.5
PNFS [41]	—	2.0	87.3
EfficientPose-A	1.3M	0.7	88.1
Hourglass [4]	25.1M	19.1	89.2
SimpleBaseline-R101 [5]	52.0M	16.5	89.1
FPD [26]	3.0M	9.0	89.0
PNFS [41]	—	9.9	89.3
EfficientPose-B	3.3M	1.5	89.3
PyraNet [23]	28.1M	21.3	89.6
DU-Net [25]	7.9M	—	89.5
DU-Net [25]	15.9M	—	89.9
SimpleBaseline-R152 [5]	68.6M	21.0	89.6
HRNet-W32 [6]	28.5M	9.5	90.3
EfficientPose-C	5.0M	2.0	89.5

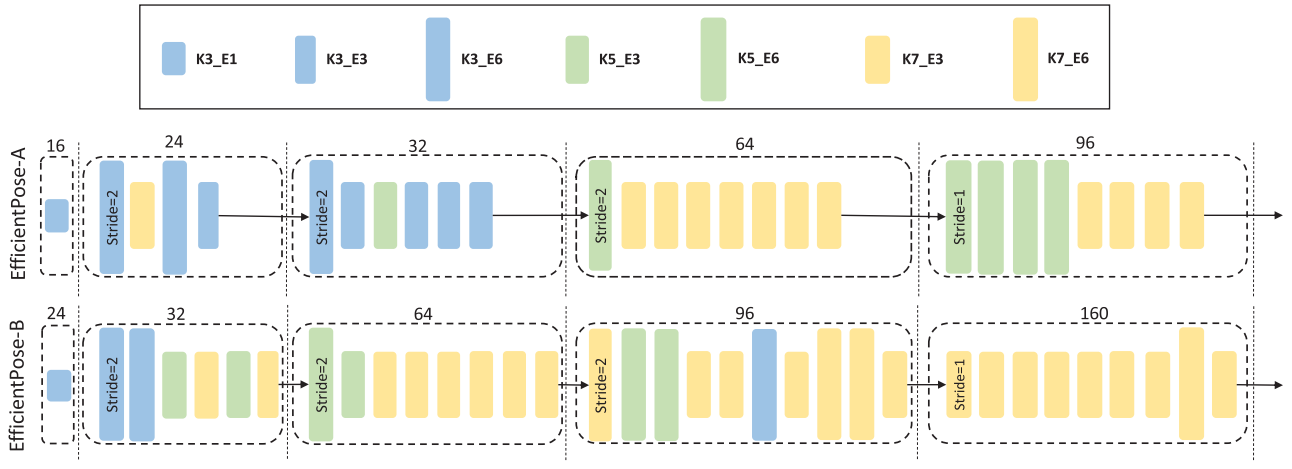

Fig. 4 Architectures of EfficientPose-A and -B. MBConv blocks with diverse kernel sizes and expansion ratios are shown as colored rectangles. "Kx_Ey" denotes a block with kernel size x and expansion ratio y. Blocks in the same stage (with the same width) are contained in dashed boxes.

close to that of the latest SOTA networks, while requiring only only 9.5% FLOPs of SimpleBaseline (ResNet152), and 21.1% of HRNet-W32 [6]. Its latency is only 27.34 ms, 44.8% of that of SimpleBaseline (ResNet152) (61.08 ms), and 49.3% of HRNet-W32 [6] (55.47 ms). The architectures found are shown in Fig. 4, while GPU latency is visualized in Fig. 5.

4.1.3 Comparisons with different backbone networks

To further demonstrate the performance of our discovered lightweight backbones, we compare our networks to others with only backbone networks changed, including manually engineered networks [11, 38] and NAS networks [17]. We train all compared networks under the same training settings and hyperparameters. We only perform 4 down-sampling operations in the comparative models for fairness.

Comparative results are shown in Table 3. In the first group, the models are all constructed with MBConv blocks [38]. EfficientPose-A achieves the highest PCKh@0.5 with the fewest FLOPs. Furthermore, compared to the large model ResNet-50 [11], EfficientPose-A achieves similar performance with 1/14.9 the number of FLOPs.

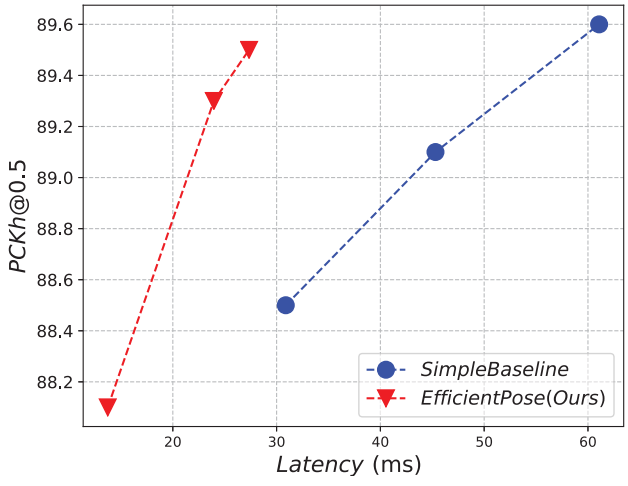

Fig. 5 GPU latency for EfficientPose and SimpleBaseline on the MPII val dataset. EfficientPose shows a far better trade-off between accuracy and latency.

Table 3 Comparisons to different backbone networks on the MPII validation set

Backbone	Params	GFLOPs	PCKh@0.5
MobileNetV2 [38]	1.5M	0.73	86.63
Proxyless(GPU) [17]	4.5M	1.43	86.26
Proxyless(CPU) [17]	1.8M	1.05	87.54
Proxyless(Mobile) [17]	1.9M	0.91	87.29
ResNet-50 [11]	25.6M	9.70	88.02
Random Search	1.4M	0.62	87.50
EfficientPose-A	1.3M	0.67	88.10

4.2 Generalizability on COCO

We applied our models discovered on MPII to COCO [8]. Only the output dimension of the final 1×1 convolution is adjusted as the number of keypoints changes. COCO contains about 200k images with about 250k person instances. We adopt the same data augmentation strategy as HRNet [6]. The input size is set to 192×256. The whole training process takes 240 epochs using the Adam optimizer with a batch size of 128. We use the warm-up strategy in the first 500 iterations to linearly increase the learning rate to 10^{-3}, and then the learning rate decays by 0.1 at 200k and 240k iterations respectively.

As shown in Table 4, our EfficientPose networks achieve high AP with fewest FLOPs on COCO. Using nearly 500 MFLOPs, the EfficientPose-A network achieves a comparable result to those of Hourglass [4], CPN [9], and LPN [42]. EfficientPose-B surpasses both LPN [42] networks and SimpleBaseline-R50 using only 1.1 GFLOPs. EfficientPose-C also obtains a high AP with far fewer FLOPs. The effectiveness of EfficientPose networks is demonstrated on the COCO test set in Table 5.

4.3 Specialization on different hardware

We specialize our EfficientPose network design on three different model cost benchmarks: FLOPs, GPU latency, and CPU latency. We achieve this by changing the cost term in the loss function (Eq. (6)) of the backbone architecture search. The GPU latency

Table 4 Generalizability on the COCO validation set

Method	Pretrain	Params	GFLOPs	AP
Hourglass [4]	N	25.1M	14.3	66.9
CPN [9]	Y	102.0M	6.2	68.6
LPN-50 [42]	N	2.9M	1.0	68.9
EfficientPose-A	N	1.3M	0.5	66.5
SimpleBaseline-R50 [5]	Y	34.0M	8.9	70.4
LPN-101 [42]	N	5.3M	1.4	70.2
LPN-152 [42]	N	7.4M	1.8	70.8
EfficientPose-B	N	3.3M	1.1	71.1
SimpleBaseline-R101 [5]	Y	52.0M	12.4	71.4
SimpleBaseline-R152 [5]	Y	68.6M	15.7	72.0
HRNet-W32 [6]	N	28.5M	7.1	73.4
MSPN [10]	Y	—	4.4	71.5
EfficientPose-C	N	5.0M	1.6	71.3

Table 5 Generalizability on the COCO test set

Method	Backbone	Input size	Params	GFLOPs	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
Mask-RCNN [43]	ResNet-50-FPN	—	—	—	63.1	87.3	68.7	57.8	71.4	—
G-RMI [44]	ResNet-101	353 × 257	42.6M	57.0	64.9	85.5	71.3	62.3	70.0	69.7
G-RMI + extra data [44]	ResNet-101	353 × 257	42.6M	57.0	68.5	87.1	75.5	65.8	73.3	73.3
CFN [45]	—	384 × 288	—	—	72.6	86.1	69.7	78.3	64.1	—
RMPE [23]	PyraNet	320 × 256	28.1M	26.7	72.3	89.2	79.1	68.0	78.6	—
CPN [9]	ResNet-Inception	384 × 288	—	—	72.1	91.4	80.0	68.7	77.2	78.5
SimpleBaseline [5]	ResNet-50	256 × 192	34.0M	8.9	70.0	90.9	77.9	66.8	75.8	75.6
SimpleBaseline [5]	ResNet-152	256 × 192	68.6M	15.7	71.6	91.2	80.1	68.7	77.2	77.3
HRNet-W32 [6]	HRNet-W32	384 × 288	28.5M	16.0	74.9	92.5	82.8	71.3	80.9	80.1
HRNet-W48 [6]	HRNet-W48	384 × 288	63.6M	32.9	75.5	92.5	83.3	71.9	81.5	80.5
LPN [42]	ResNet-50	256 × 192	2.9M	1.0	68.7	90.2	76.9	65.9	74.3	74.5
LPN [42]	ResNet-101	256 × 192	5.3M	1.4	70.0	90.8	78.4	67.2	75.4	75.7
LPN [42]	ResNet-152	256 × 192	7.4M	1.8	70.4	91.0	78.9	67.7	76.0	76.2
PNFS [41]	MobileNet-V2	256 × 192	6.1M	4.0	67.4	89.0	73.7	63.3	74.3	73.1
PNFS [41]	ResNet-50	256 × 192	27.5M	11.4	70.9	90.4	77.7	66.7	78.2	76.6
EfficientPose-B	NAS searched	256 × 192	3.3M	1.1	70.5	91.1	79.0	67.3	76.2	76.1
EfficientPose-C	NAS searched	256 × 192	5.0M	1.6	70.9	91.3	79.4	67.7	76.5	76.5

is measured on one Tesla V100 GPU with batch size 32 and the CPU latency is measured on 1 Intel(R) Core(TM) i7-8700K CPU with a batch size of 1. For each cost benchmark, we search for two networks and show results in Fig. 6. This experiment shows that our pose estimation framework can be efficient on diverse hardware platforms. The specialization process can be easily performed by building different lookup tables on the cost benchmarks for predicting model cost. Though FLOPs are widely used to evaluate the cost of a model, the results in Fig. 6 and other recent works [17, 36] demonstrate that FLOPs are not well correlated with real inferencing speed due to differences in hardware platforms. The ability to specialize models to a target device is crucial for real applications.

4.4 Ablation studies

4.4.1 Effectiveness of SIC module

We visualize the feature maps in the network to study the effect of the SIC module in Fig. 9. We find that a checkerboard pattern of artifacts [28] exists in both the feature maps after the transposed convolutions in the network trained without SIC and the feature maps before SIC in EfficientPose networks. The SIC module eliminates the checkerboard pattern in the feature maps obtained by transposed convolutions and makes the field of interest more concentrated, which contributes to the final prediction. We provide ablation study results for SIC in Table 6. The SIC module provides evident accuracy improvement in both the manually designed network MobileNetV2 and the EfficientPose networks.

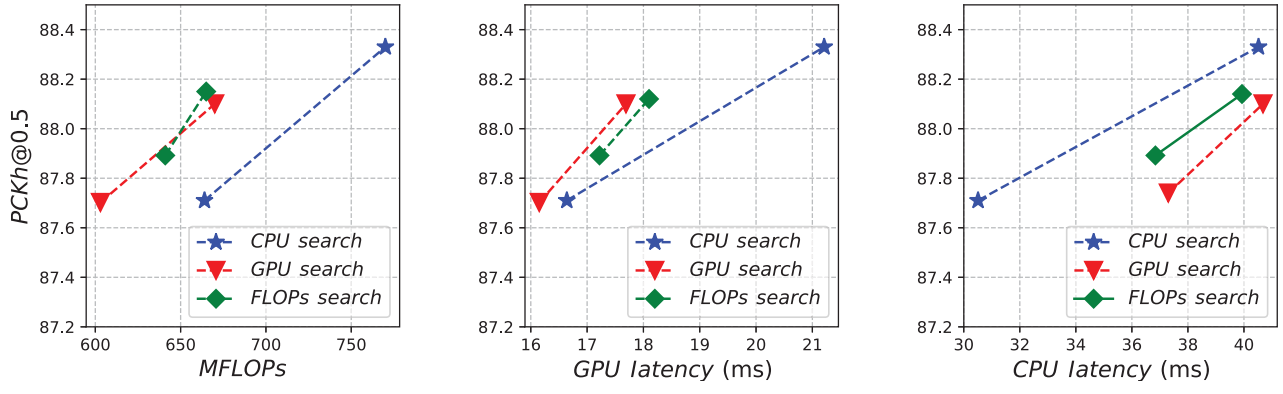

Fig. 6 Specialization results of different model cost benchmarks on the MPII validation set. Networks optimized with different benchmarks are shown in different colors.

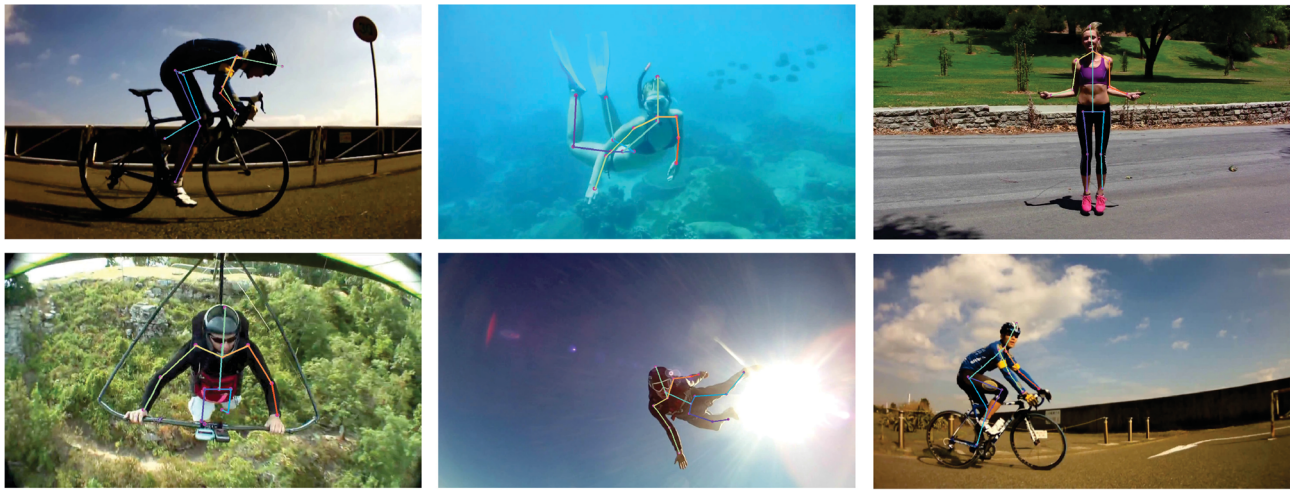

Fig. 7 Results on the MPII validation set.

Fig. 8 Results on the COCO validation set.

Table 6 Ablation study, SIC module

Model	SIC	PCKh@0.5
MobileNetv2 [38]		86.34
	✓	$86.63_{\uparrow 0.29}$
EfficientPose-A		87.62
	✓	$88.10_{\uparrow 0.48}$
EfficientPose-B		88.55
	✓	$89.27_{\uparrow 0.72}$
EfficientPose-C		88.61
	✓	$89.49_{\uparrow 0.88}$

4.4.2 Efficiency of slim transposed convolutions

We set the widths of the two transposed convolutions in the head to 64 and 32 respectively; these have always been set larger in previous work, e.g., 256 in SimpleBaseline [5]. To further demonstrate the efficiency of our slim transposed convolutions, we compare use of larger width settings for the two transpose convolutions in the head. As shown in Table 7, when we enlarge the widths to 64 and 64, the FLOPs used by the head increase to 134M but no

Table 7 Comparison to larger width settings for the transposed convolutions. TConv denotes the transposed convolution

Width		MFLOPs		PCKh@0.5
TConv1	TConv2	Backbone	Head	
64	32	405	264	88.100
64	64		369	88.095
96	96		755	87.918

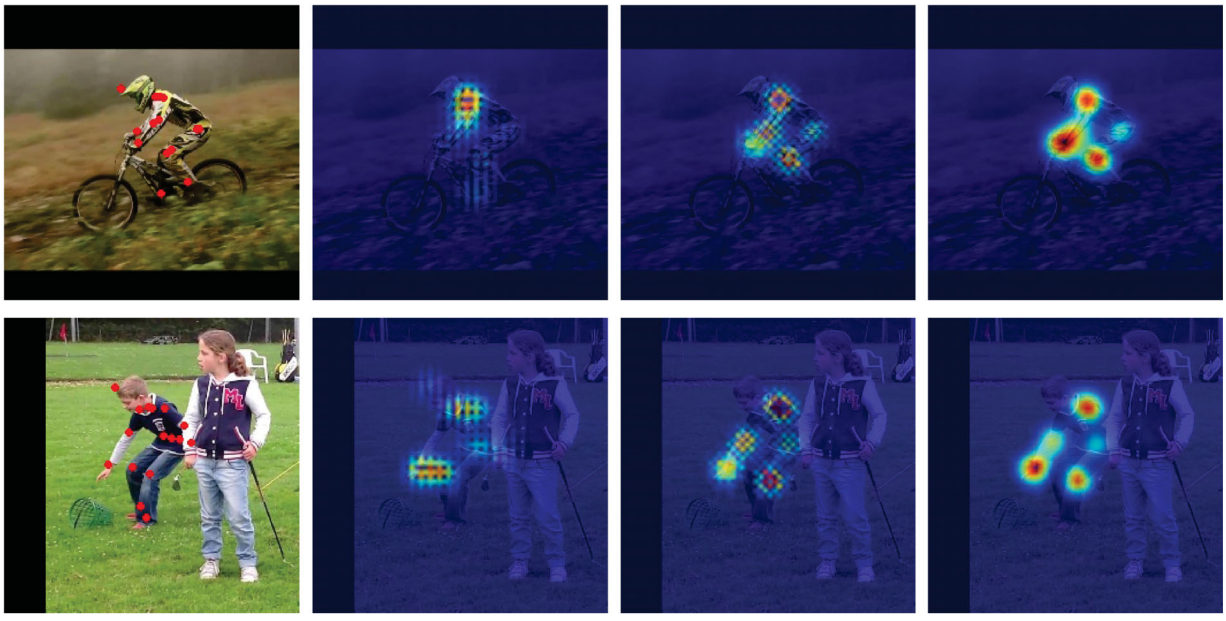

Fig. 9 Feature maps in the network. (a) Ground truth. (b) Feature maps after transposed convolutions in the network trained without SIC. (c) Feature maps before SIC in the EfficientPose network. (d) Feature maps after SIC in the EfficientPose network.

accuracy improvement is obtained. When we set the widths larger, to 96 and 96, the FLOPs used increase to 520M and a worse PCKh@0.5 results. It is worth noting the FLOPs required by the (96, 96) head is much larger than for the backbone. We attribute the performance degradation to overfitting when the head is too heavy with a lightweight backbone.

4.4.3 Studies on efficient transposed convolutions

We study lighter convolution modules for the transposed convolutions, the separable depthwise convolution in MobileNetV1 [12] (MBV1), and the inverted residual block in MobileNetV2 [38] (MBV2). As shown in Table 8, we find that the MBV1 style module decreases the FLOPs to 181M with acceptable accuracy decay, which could be an alternative option for the efficient head .

4.4.4 Comparisons with random search

We perform a random search experiment, which is a key baseline for evaluating the effectiveness of the NAS method [46]. We randomly sample 50 architectures and train each one for 5 epochs to evaluate the validation performance. Finally, we select the best-performing one and train it with the same settings as our EfficientPose. The total cost of random search is the same as ours. The results are shown in Table 3. Our EfficientPose-A shows an evident advantage over the random searched one.

Table 8 Other efficient transposed convolutions

Transposed Conv	MFLOPs	PCKh@0.5
Plain	669	88.10
MBV1 style	469	87.91
MBV2 style	600	87.94

5 Conclusions

In this paper, we have proposed a framework for efficient human pose estimation. We use the differentiable NAS method to automatically customize the backbone network for pose estimation and greatly reduce the computational cost. We further give an efficient head network which includes both slim transposed convolutions and a spatial information correction module to improve prediction performance with negligible FLOPs or latency increases. The proposed EfficientPose networks achieve similar accuracies to other SOTA methods with far less computational cost.

Acknowledgements

This work was in part supported by National Natural Science Foundation of China (NSFC) (Nos. 61733007 and 61876212) and Zhejiang Lab (No. 2019NB0AB02).

References

[1] Yang, Y.; Ramanan, D. Articulated pose estimation with flexible mixtures-of-parts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1385–1392, 2011.

[2] Pishchulin, L.; Andriluka, M.; Gehler, P.; Schiele, B. Poselet conditioned pictorial structures. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 588–595, 2013.

[3] Toshev, A.; Szegedy, C. DeepPose: Human pose estimation via deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1653–1660, 2014.

[4] Newell, A.; Yang, K. Y.; Deng, J. Stacked hourglass networks for human pose estimation. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9912*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 483–499, 2016.

[5] Xiao, B.; Wu, H. P.; Wei, Y. C. Simple baselines for human pose estimation and tracking. In: *Computer Vision – ECCV 2018. Lecture Notes in Computer Science, Vol. 11210*. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 472–487, 2018.

[6] Sun, K.; Xiao, B.; Liu, D.; Wang, J. D. Deep high-resolution representation learning for human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5686–5696, 2019.

[7] 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, 3686–3693, 2014.

[8] 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: *Computer Vision – ECCV 2014. Lecture Notes in Computer Science, Vol. 8693*. Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 740–755, 2014.

[9] Chen, Y. L.; Wang, Z. C.; Peng, Y. X.; Zhang, Z. Q.; Yu, G.; Sun, J. Cascaded pyramid network for multi-person pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7103–7112, 2018.

[10] Li, W. B.; Wang, Z. C.; Yin, B. Y.; Peng, Q. X.; Su, J. Rethinking on multi-stage networks for human pose estimation. *arXiv preprint* arXiv:1901.00148, 2019.

[11] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016.

[12] Howard, A. G.; Zhu, M. L.; Chen, B.; Kalenichenko, D.; Adam, H. Mobilenets: Efficient convolutional neural networks formobile vision applications. *arXiv preprint* arXiv:1704.04861, 2017.

[13] Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q. V. Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8697–8710, 2018.

[14] Real, E.; Aggarwal, A.; Huang, Y. P.; Le, Q. V. Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 4780–4789, 2019.

[15] Bender, G.; Kindermans, P.; Zoph, B.; Vasudevan, V.; Le, Q. Understanding and simplifying one-shot architecture search. In: Proceedings of the 35th International Conference on Machine Learning, 549–558, 2018.

[16] Liu, H. X.; Simonyan, K.; Yang, Y. M. DARTS: Differentiable architecture search. In: Proceedings of the 7th International Conference on Learning Representations, 2019.

[17] Cai, H.; Zhu, L.; Han, S. ProxylessNAS: Direct neural architecture search on target task and hardware. In: Proceedings of the International Conference on Learning Representations, 2019.

[18] Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; Keutzer, K. Fbnet: Hardware-aware efficient convNet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10726–10734, 2019.

[19] Liu, C. X.; Chen, L. C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; Fei-Fei, L. Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 82–92, 2019.

[20] Zhang, Y.; Qiu, Z.; Liu, J.; Yao, T.; Liu, D.; Mei, T. Customizable architecture search forsemantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11633–11642, 2019.

[21] Ghiasi, G.; Lin, T. Y.; Le, Q. V. NAS-FPN: Learning scalable feature pyramid architecture for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7029–7038, 2019.

[22] Fang, J. M.; Sun, Y. Z.; Zhang, Q.; Peng, K. J.; Wang, X. G. FNA++: Fast network adaptation via parameter remapping and architecture search. In: Proceedings of the International Conference on Learning Representations, 2020.

[23] Yang, W.; Li, S.; Ouyang, W. L.; Li, H. S.; Wang, X. G. Learning feature pyramids for human pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, 1290–1299, 2017.

[24] Bulat, A.; Tzimiropoulos, G. Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. In: Proceedings of the IEEE International Conference on Computer Vision, 3726–3734, 2017.

[25] Tang, Z. Q.; Peng, X.; Geng, S. J.; Wu, L. F.; Zhang, S. T.; Metaxas, D. Quantized densely connected U-nets for efficient landmark localization. In: *Computer Vision – ECCV 2018. Lecture Notes in Computer Science, Vol. 11207*. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 348–364, 2018.

[26] Zhang, F.; Zhu, X. T.; Ye, M. Fast human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3512–3521, 2019.

[27] Wei, S. H.; Ramakrishna, V.; Kanade, T.; Sheikh, Y. Convolutional pose machines. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4724–4732, 2016.

[28] Odena, A.; Dumoulin, V.; Olah, C. Deconvolution and checkerboard artifacts. *Distill*, 2016. Available at http://doi.org/10.23915/distill.

[29] Gao, H.; Yuan, H.; Wang, Z.; Ji, S. Pixel deconvolutional networks. *arXiv preprint* arXiv:1705.06820, 2017.

[30] Wojna, Z.; Uijlings, J.; Guadarrama, S.; Silberman, N.; Chen, L. C.; Fathi, A.; Uijlings, J. The devil is in the decoder. In: Proceedings of the British Machine Vision Conference, 10.1–10.13, 2017.

[31] Sugawara, Y.; Shiota, S.; Kiya, H. Checkerboard artifacts free convolutional neural networks. *APSIPA Transactions on Signal and Information Processing* Vol. 8, e9, 2019.

[32] Tan, M. X.; Le, Q. V. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint* arXiv:1905.11946, 2019.

[33] Brock, A.; Lim, T.; Ritchie, J. M.; Weston, N. SMASH: One-shot model architecture search through HyperNetworks. In: Proceedings of the International Conference on Learning Representations, 2018.

[34] Dong, X. Y.; Yang, Y. Searching for a robust neural architecture in four GPU hours. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1761–1770, 2019.

[35] Xu, Y. H.; Xie, L. X.; Zhang, X. P.; Chen, X.; Xiong, H. K. PC-DARTS: Partial channel connections for memory-efficient differentiable architecture search. In: Proceedings of the International Conference on Learning Representations, 2019.

[36] Tan, M. X.; Chen, B.; Pang, R. M.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q. V. MnasNet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2815–2823, 2019.

[37] Gong, X. Y.; Chen, W. Y.; Jiang, Y. F.; Yuan, Y.; Wang, Z. Y. AutoPose: Searching multi-scale branch aggregation for pose estimation. *arXiv preprint* arXiv:2008.07018, 2020.

[38] Sandler, M.; Howard, A.; Zhu, M. L.; Zhmoginov, A.; Chen, L. C. MobileNetV2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4510–4520, 2018.

[39] Fang, J. M.; Sun, Y. Z.; Zhang, Q.; Li, Y.; Wang, X. G. Densely connected search space for more flexible neural architecture search, In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10625–10634, 2020.

[40] Tang, W.; Yu, P.; Wu, Y. Deeply learned compositional models for human pose estimation. In: *Computer Vision – ECCV 2018. Lecture Notes in Computer Science, Vol. 11207*. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 197–214, 2018.

[41] Yang, S.; Yang, W. K.; Cui, Z. Pose neural fabrics search. *arXiv preprint* arXiv:1909.07068, 2019.

[42] Zhang, Z.; Tang, J.; Wu, G. Simple and lightweight human pose estimation. *arXiv preprint* arXiv:1911.10346, 2019.

[43] He, K. M.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2980–2988, 2017.

[44] Papandreou, G.; Zhu, T.; Kanazawa, N.; Toshev, A.; Tompson, J.; Bregler, C.; Murphy, K. Towards accurate multi-person pose estimation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3711–3719, 2017.

[45] Huang, S. L.; Gong, M. M.; Tao, D. C. A coarse-fine network for keypoint localization. In: Proceedings of the IEEE International Conference on Computer Vision, 3047–3056, 2017.

[46] Ottelander, T. D.; Dushatskiy, A.; Virgolin, M.; Bosman, P. A. N. Local search is a remarkably strong baseline for neural architecture search. *arXiv preprint* arXiv:2004.08996, 2020.

Wenqiang Zhang is a master student in the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan. His research interests include pose estimation and neural architecture search.

Jiemin Fang received his B.E. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology in 2018. He is currently a Ph.D. candidate at the Institute of Artificial Intelligence and School of Electronic Information and Communications, Huazhong University of Science and Technology. His research interests include AutoML and efficient deep learning.

Xinggang Wang received his B.S. and Ph.D. degrees in electronics and information engineering from Huazhong University of Science and Technology, in 2009 and 2014, respectively. He is currently an associate professor with the School of Electronic Information and Communications, HUST. His research interests include computer vision and machine learning.

Wenyu Liu received his B.S. degree in computer science from Tsinghua University, Beijing, China, in 1986, and his M.S. and Ph.D. degrees, both in electronics and information engineering, from Huazhong University of Science and Technology (HUST), in 1991 and 2001, respectively. He is now a professor and associate dean of the School of Electronic Information and Communications, HUST. His current research areas include computer vision, multimedia, and machine learning.