Lite Pose: Efficient Architecture Design for 2D Human Pose Estimation

Yihan Wang[1*] Muyang Li[2] Han Cai[3] Weiming Chen[3] Song Han[3]
[1]Tsinghua University [2]Carnegie Mellon University [3]Massachusetts Institute of Technology
https://tinyml.mit.edu

Abstract

Pose estimation plays a critical role in human-centered vision applications. However, it is difficult to deploy state-of-the-art HRNet-based pose estimation models on resource-constrained edge devices due to the high computational cost (more than 150 GMACs per frame). In this paper, we study efficient architecture design for real-time multi-person pose estimation on edge. We reveal that HRNet's high-resolution branches are redundant for models at the low-computation region via our ***gradual shrinking*** *experiments. Removing them improves both efficiency and performance. Inspired by this finding, we design* ***LitePose****, an efficient single-branch architecture for pose estimation, and introduce two simple approaches to enhance the capacity of LitePose, including* ***fusion deconv head*** *and* ***large kernel conv****. On mobile platforms, LitePose reduces the latency by up to 5.0× without sacrificing performance, compared with prior state-of-the-art efficient pose estimation models, pushing the frontier of real-time multi-person pose estimation on edge. Our code and pre-trained models are released at https://github.com/mit-han-lab/litepose.*

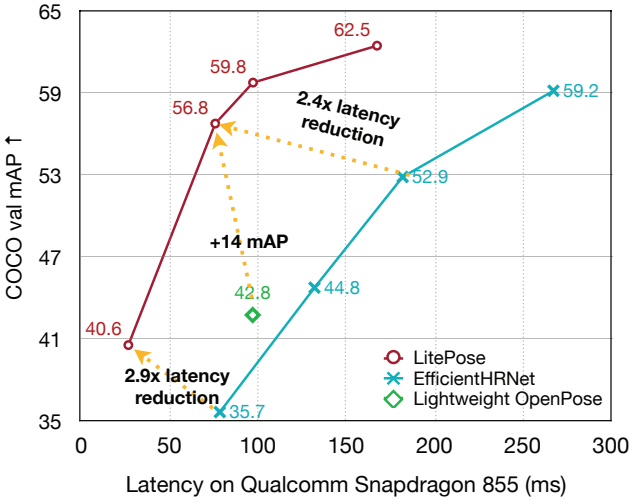


Figure 1. LitePose provides up to 2.9× latency reduction compared to EfficientHRNet [36] on Qualcomm Snapdragon 855 while achieving higher mAP on COCO. Compared with Lightweight OpenPose [39], LitePose obtains 14% higher mAP on COCO with lower latency.

1. Introduction

Human pose estimation aims to predict each person's keypoint positions from an image. It is a critical technique for many vision applications that require understanding human behavior. Typical human pose estimation models can be categorized into two paradigms: top-down and bottom-up. The top-down paradigm [7, 10, 15, 23, 38, 44, 47, 51] first detects people via an extra person detector and then performs single-person pose estimation for each detected person. In contrast, the bottom-up paradigm [5, 8, 11, 21, 22, 24, 37, 38, 40, 41] first predicts identity-free keypoints and then groups them into persons. As the bottom-up paradigm does not involve an extra person detector and does not require repeatedly running the pose estimation model for each person in the image, it is more suitable for real-time multi-person pose estimation on edge.

However, existing bottom-up pose estimation models [5, 8, 11, 21, 22, 24, 37, 38, 40, 41] mainly focus on the high-computation region. For instance, HigherHRNet [8] achieves its best performance on the CrowdPose dataset [26] with more than 150GMACs, which is prohibitive for edge devices. It is of great importance to design models with low computational cost while maintaining good performances.

In this paper, we study efficient architecture design for bottom-up human pose estimation. Previous study [8, 11] in the high-computation region suggests that maintaining the high-resolution representation plays a critical role in achieving good performances for bottom-up pose estimation. However, it is unclear whether this still holds for models in the low-computation region. To answer this question, we build a "bridge" between the representative multi-branch architecture, HigherHRNet [8], and the single-branch archi-

* work done while interning at MIT HAN Lab

tecture by **gradual shrinking** (Figure 2). We surprisingly find that the performance improves as we shrink the depth of high-resolution branches for models in the low-computation region (Figure 3). Inspired by this finding, we design a single-branch architecture, *LitePose*, for efficient bottom-up pose estimation. In *LitePose*, we use a modified MobileNetV2[1] [43] backbone with two important improvement to efficiently handle the scale variation problem in the single-branch design: **fusion deconv head** and **large kernel conv**. The fusion deconv head removes the redundant refinement in high-resolution branches and therefore allows scale-aware multi-resolution fusion in a single-branch way (Figure 6). Meanwhile, different from image classification, we find large kernel convs provide a much more prominent improvement in bottom-up pose estimation (Figure 7). Finally, we apply Neural Architecture Search (NAS) to optimize the model architecture and choose appropriate input resolution.

Extensive experiments on CrowdPose [26] and COCO [28] demonstrate the effectiveness of *LitePose*. On CrowdPose [26], *LitePose* achieves $2.8\times$ MACs reduction and up to $5.0\times$ latency reduction with better performance. On COCO [28], LitePose obtains $2.9\times$ latency reduction compared with EfficientHRNet [36] while providing better performances.

We summarize our contributions as follows:

1. We design **gradual shrinking** experiments, revealing that the high-resolution branches are redundant for models in the low-computation region.
2. We propose *LitePose*, an efficient architecture for bottom-up pose estimation. We also introduce two techniques to enhance the capacity of LitePose, including **fusion deconv head** and **large kernel conv**.
3. Extensive experiments on two benchmark datasets, Microsoft COCO [28] and CrowdPose [26] demonstrate the effectiveness of our method: LitePose achieves up to $2.8\times$ MACs reduction and up to $5.0\times$ latency reduction compared with state-of-the-art HRNet-based models.

2. Related Work

2D Human Pose Estimation. 2D human pose estimation aims at localizing human anatomical keypoints (*e.g.*, elbow, wrist) or parts. There are two main frameworks: the top-down framework and the bottom-up framework. Top-down methods [7, 10, 15, 23, 38, 44, 47, 51] perform single-person pose estimation by firstly detecting each person from the image. On the contrary, bottom-up methods [5, 8, 11, 21, 22, 24, 37, 38, 40, 41] directly predict keypoints of each person in an end-to-end manner. Typical bottom-up methods consist of two steps: predicting keypoint heatmaps and then grouping the detected keypoints into persons. Among these approaches, HRNet-based multi-branch architectures [8, 11] provide state-of-the-art results. They design a multi-branch architecture to allow multi-resolution fusion, which has been proven effective in solving scale variation problems for bottom-up pose estimation. However, all these approaches are too computationally intensive (most >150GMACs) to be deployed on edge devices. In this work, we focus on the bottom-up framework for efficiency. Following state-of-the-art HRNet-based approaches [8], we use associative embedding [37] for grouping.

Model Acceleration. Apart from designing efficient models directly [20, 34, 35, 43, 50, 55], another approach for model acceleration is to compress existing large models. Some methods aim at pruning the redundancy inside connections and convolution filters [13, 14, 18, 27, 32, 48]. Meanwhile, some other methods focus on quantizing the network [9, 25, 46, 57]. Besides, several AutoML methods have also been proposed to automate the model compression and acceleration [17, 33, 46, 52]. Recently, Yu *et al*. design LiteHRNet [53] for top-down pose estimation, while we focus on the bottom-up paradigm. Neff *et al*. propose EfficientHRNet [36] for the efficient bottom-up pose estimation. They apply the compound scaling idea in EfficientNet [45] to HigherHRNet [8] and achieve $1.5\times$ MACs reduction. However, their method still faces drastic performance degradation when the computational constraint becomes tighter. In this work, we push the MACs reduction ratio to $5.1\times$ and achieves up to $5.0\times$ latency reduction on mobile platforms compared to EfficientHRNet.

Neural Architecture Search. Neural Architecture Search (NAS) has achieved great success on large-scale image classification tasks [2, 29, 30, 58]. Automatically designed models significantly outperform hand-crafted ones. To make the search process more efficient, researchers proposed one-shot NAS methods [1, 3, 4, 12, 19, 31, 49] in which different sub-networks share the same set of weights. To further explore the potential of our proposed architecture, we apply the *once-for-all* [3] approach to automatically prune the redundancy inside channels and select the appropriate input size. Compared to the manually designed models trained from scratch, our searched models achieve prominent up to $+3.6$AP improvement.

3. Rethinking the Efficient Design Space

Multi-branch networks have achieved great success on the bottom-up pose estimation task. Their representative, HigherHRNet [8], uses multi-branch architecture to help

[1]Our method can also be combined with other backbones. We choose MobileNetV2 as it only contains basic operations (1×1 conv, depthwise conv, Relu6) that are well-supported on most edge platforms.

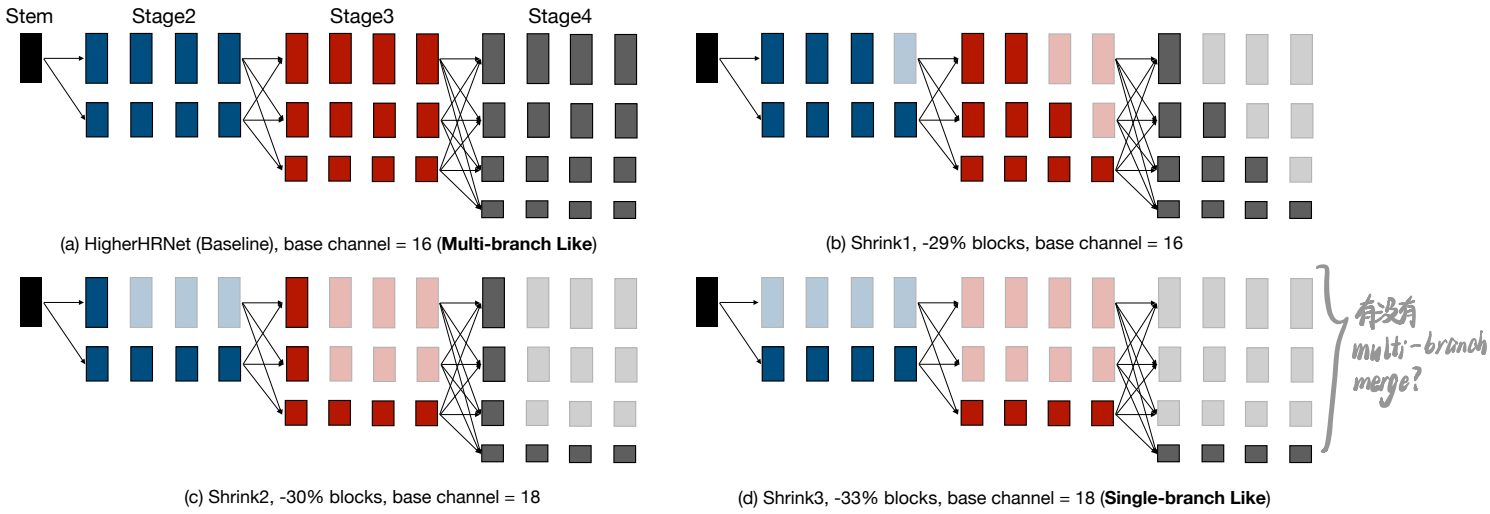


Figure 2. Four architecture configurations in the **gradual shrinking** expriment. We use *HigherHRNet* [8] as the baseline for comparison. Removed blocks are shown in transparent. The network becomes increasingly close to the single-branch architecture from *Baseline* to *Shrink3*. To ensure different architecture configurations have similar MACs, we increase the base channel from 16 to 18 for *Shrink2* and *Shrink3*.

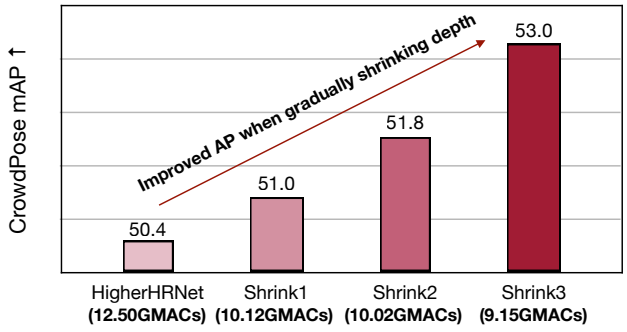


Figure 3. The performance improves as we gradually shrink the high-resolution branches of *HigherHRNet-W16*.

fuse multi-resolution features, which significantly alleviate the scale variation problem. Benefiting from this, multi-branch architectures outperform single-branch architectures and obtain state-of-the-art results. But there remains a problem in most of these methods [8, 11, 38, 40] that they achieve their best performance with more than 150GMACs. The comparisons among methods are also mostly conducted with such high computation. Towards real-world edge applications, studies on efficient human pose estimation with lower computation are of high priority. In this section, we first introduce HRNet-based multi-branch architectures and how they cope with the scale variation problem. Then we point out the redundancy in high-resolution branches by **gradual shrinking** in computationally limited cases. Based on this observation, we propose the **fusion deconv head**, which removes the redundant refinement in high-resolution branches and therefore handles the scale variation problem in an efficient way. On the other hand, we empirically find the large kernels provide much more prominent improvement on the pose estimation task compared with the image classification task. Extensive experiments and ablation studies show the effectiveness of our method and reveal a fact that properly designed single-branch architectures can achieve better performance and lower latency.

3.1. Scale-Aware Multi-branch Architectures

Scale-Awareness. The multi-branch design aims to alleviate the scale variation problem in bottom-up pose estimation. Since we need to predict the joint coordinates of all persons in an image, it is usually hard for single-branch architectures to recognize small persons and distinguish close joints from final low-resolution features, as shown in Figure 5(b). The high-resolution features introduced by multi-branch architectures, however, can reserve more detailed information and therefore help neural networks better capture small persons and discriminate close joints.

Mechanism. As shown in Figure 2, the main part of HRNet-based multi-branch architecture [8, 11] consists of 4 stages. In stage n (we regard stem as stage 1 here), there are n branches handling n different input feature maps with different resolutions, respectively. When processing input features, each branch first refines its own input feature respectively, then exchanges information among branches to obtain multi-scale information.

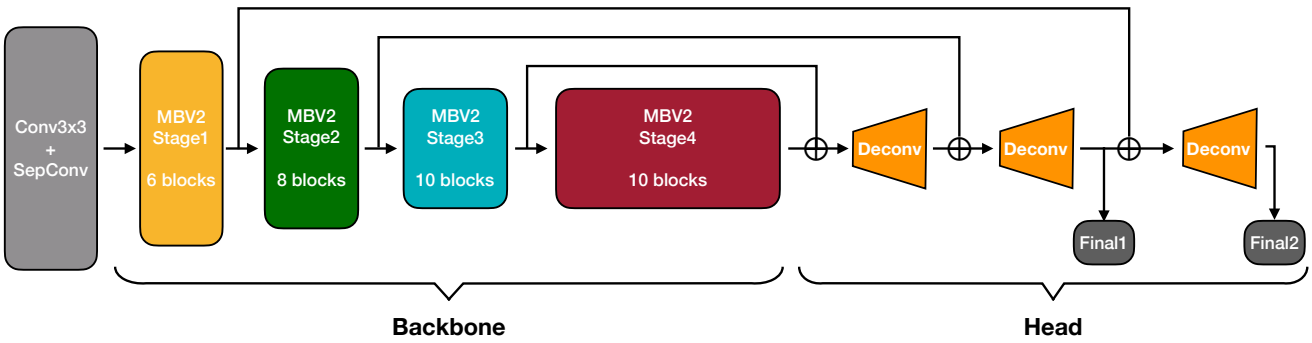


Figure 4. The architecture of *LitePose*. *LitePose* consists of the backbone and the fusion deconvolution head. ⊕ means the "concatenate" operation. The final convs are used for multi-resolution supervision following [8].

3.2. Redundancy in High-Resolution Branches

However, when focusing on the performance with lower computation, we find the multi-branch architecture may not be the most efficient choice. In this section, we propose a method called **gradual shrinking** to reveal the redundancy in the high-resolution branches of the multi-branch architecture. As shown in Figure 2 and Figure 3, by gradually shrinking the depth of high-resolution branches, the multi-branch network behaves increasingly like a single-branch network. However, the performance does not degrade even improve.

Gradual Shrinking. To reveal the redundancy inside the HRNet-based multi-branch architecture [8, 11], we design a gradual shrinking experiment on the branches in each stage. Let $A_n = [a_1, \ldots, a_n]$ denote the number of blocks used to refine features for each branch (a_i stands for the number of blocks in branch i) in stage n before fusion. Here, branch i processes feature maps with higher resolution than branch $i+1$. Then we can define the configuration of the whole multi-branch architecture as $A = \{A_1, A_2, A_3, A_4\}$. We say $A'_i = [a'_1, \ldots, a'_i]$ is shrunk from $A_i = [a_1, \ldots, a_i]$ if $\forall j \in \{1, \ldots, i\}, a'_j \leq a_j$. For convenience, we denote this as $A'_i \leq A_i$. A configuration A' is said to be shrunk from A (*i.e.*, $A' \leq A$) if $\forall i \in \{1, 2, 3, 4\}$, $A'_i \leq A_i$. With the aforementioned notations, **gradual shrinking** means that we construct a sequence of configurations $[C_1, \ldots, C_m]$ s.t. $C_{i+1} \leq C_i, \forall i \in \{1, \ldots, m-1\}$. As shown in Figure 2 and Figure 3, we gradually shrink the depth of high-resolution branches and surprisingly find that this shrinking operation even helps improve the performance. Meanwhile, the gradual shrinking process makes the whole network increasingly similar to a single-branch network, which provides strong evidence that the single-branch architecture is more suitable for the efficient architecture design on the bottom-up pose estimation task. To make the gradual shrinking process clearer, we list the four configurations we use in detail below:

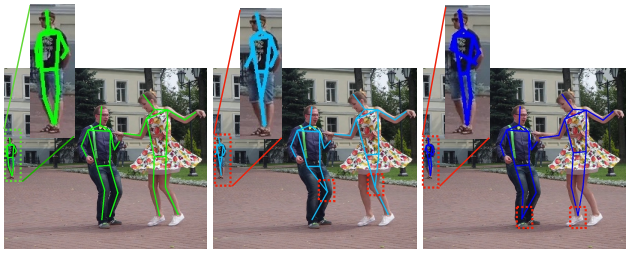
- Baseline: $C_1 = \{[4], [4, 4], [4, 4, 4], [4, 4, 4, 4]\}$, 12.5GMACs, base channel=16
- Shrink 1: $C_2 = \{[4], [3, 4], [2, 3, 4], [1, 2, 3, 4]\}$, 10.1GMACs, base channel=16
- Shrink 2: $C_3 = \{[4], [1, 4], [1, 1, 4], [1, 1, 1, 4]\}$, 10.0GMACs, base channel=18
- Shrink 3: $C_4 = \{[4], [0, 4], [0, 0, 4], [0, 0, 0, 4]\}$, 9.2GMACs, base channel=18

3.3. Fusion Deconv Head: Remove the Redundancy

Though we have shown the redundancy in the multi-branch architecture above, its strong capability of handling the scale variation problem is still remarkable. Can we combine this feature into our design while keeping the merits of single-branch architecture (*e.g.*, high efficiency)? To achieve this goal, we propose the fusion deconvolution layers as our final prediction head. To be specific, as shown in Figure 4 and 6(b), we **directly** (*i.e.* without any refinement) utilize the low-level high-resolution features generated by previous stages for deconvolution and final prediction layers. On the one hand, our *LitePose* uses the single-branch architecture as our backbone, which benefits from the low-latency characteristic. On the other hand, directly using low-level high-resolution features avoids the redundant refinement in multi-branch HR fusion modules. Therefore, LitePose inherits the advantages from both single-branch design and multi-branch design in an efficient way. In Figure 6(a) and Figure 5, we show the strength of our fusion deconvolution head. With a negligible computational cost increase, we obtain a significant performance improvement (+7.6AP).

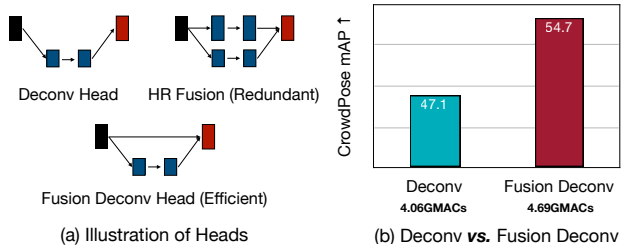
3.4. Mobile Backbone with Large Kernel Convs

Several papers [20, 35, 43, 55] have studied efficient architectures under tight computational constraints on the image classification task. As shown in Figure 4, we use a modified MobileNetV2 [43] architecture as the backbone in



(a) 7x7 kernel, w/ fusion, 54.7AP (b) 7x7 kernel, w/o fusion, 47.1AP (c) 3x3 kernel, w/ fusion, 41.7AP

Figure 5. Visualization of models with/without larger kernel convs and fusion deconv head. *LitePose* can better recognize small persons and distinguish close joints with larger kernel convs and fusion deconv head.



(a) Illustration of Heads (b) Deconv ***vs.*** Fusion Deconv

Figure 6. Unlike conventional single-branch deconv head (from the black block to the red block), our fusion deconv head takes the advantage of HR fusion module and remove the high-resolution redundant refinement blocks. It achieves great improvement (+7.6AP) comparing to normal deconv head with minor computation increase.

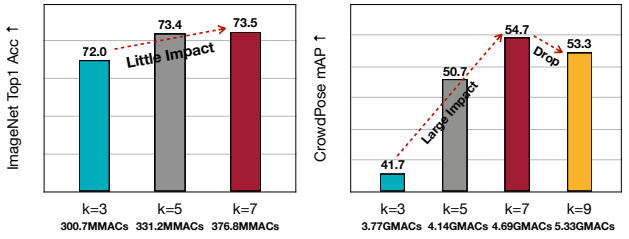


Figure 7. k represents the kernel size. Increasing the kernel size provides moderate performance improvement for image classification but making a big difference for pose estimation. Specifically, increasing the kernel size from 3 to 7 provides 13% mAP improvement on CrowdPose.

LitePose. Following [54], we make a minor modification on the original MobileNetV2 [43] backbone by removing the final down-sampling stage. Too many down-sampling layers will cause essential information loss, which is harmful to the high-resolution output of the pose estimation task.

To further alleviate the scale variation problem, we introduce large kernels into our efficient architecture design. Unlike the traditional image classification task, this modification plays a much more important role in our proposed

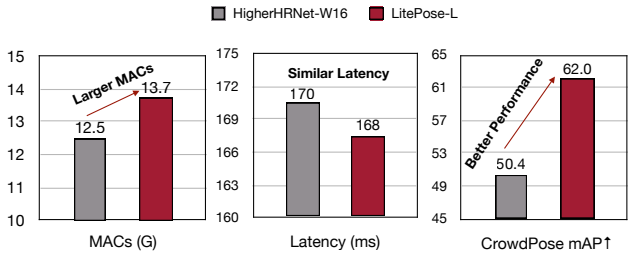


Figure 8. Compared with the **multi-branch** *HigherHRNet-W16* [8], **single-branch** *LitePose-L* executes with higher parallelism. Therefore it achieves much better performance and similar latency on Qualcomm Snapdragon 855 with even higher MACs.

MobileNetV2-based [43] backbone. In Figure 7, we show the performance comparisons among models with kernel sizes 3,5,7 (and 9 only for pose estimation) on both the image classification and the pose estimation task. With a similar computational cost increase (about +25%), the performance gain on the pose estimation task (+13.0AP) is much more significant than on the image classification task (+1.5% Acc). The visualization results in Figure 5 also verify our claim. However, the rule is not "the larger, the better". Too large kernels will introduce many useless parameters and nonnegligible noise, which makes the training more difficult and incurs performance degradation, demonstrated in Figure 7 for $k = 9$ case. Since we further find incorporating kernel size into the search space will severely degenerate the performance of NAS mentioned in Section 4, which may be caused by the large impact of the kernel size variation, we fix the kernel size to 7×7 in our architecture.

3.5. Single Branch, High Efficiency

Besides the performance, another important advantage of our single-branch *LitePose* is its hardware-friendly characteristic. As mentioned in ShuffleNetV2 [35], network fragmentation such as multi-branch design reduces the degree of parallelism on some hardware. Therefore, towards real-world applications, single-branch architecture is a better choice. In summary, we show the quantitative comparison results between *HigherHRNet-W16* [8] and *LitePose-L* in Figure 3. Compared with *HigherHRNet-W16* [8], *LitePose-L* not only achieves much better performance (+11.6AP), but also obtains similar latency on Qualcomm Snapdragon 855 with even with larger MACs. All these results demonstrate the high efficiency of our single-branch *LitePose*.

4. Neural Architecture Search

Existing work [5, 8, 11, 24, 38–40] on the bottom-up pose estimation task usually uses a hand-crafted (and mostly uniform) channel width across all the layers in the model and a fixed large resolution (*e.g.*, 512×512). To further explore the potential compactness of our model, in this section, we

Model	Input Size	#Params	#MACs		Latency (ms)						AP	AP^{50}	AP^{75}
					Nano		Mobile		Pi				
HigherHRNet-W48 [8]	640×640	63.8M	154.6G	–	2101	–	1532	–	12302	–	65.9	86.4	70.6
HigherHRNet-W24	512×512	14.9M	25.3G	–	330	–	289	–	1414	–	57.4	83.2	63.2
EfficientHRNet-H_{-1} [36]	480×480	13.0M	14.2G	(1.8×)	283	(1.2×)	267	(1.1×)	1229	(1.2×)	56.3	81.3	59.0
LitePose-S (Ours)	448×448	2.7M	5.0G	(5.1×)	97	(3.4×)	76	(3.8×)	420	(3.4×)	58.3	81.1	61.8
HigherHRNet-W16	512×512	7.2M	12.5G	–	172	–	170	–	898	–	50.4	78.4	54.5
EfficientHRNet-H_{-3} [36]	416×416	5.3M	4.3G	(2.9×)	111	(1.5×)	132	(1.3×)	544	(1.7×)	46.1	79.3	48.3
LitePose-XS (Ours)	256×256	1.7M	1.2G	(10.4×)	22	(7.8×)	27	(6.3×)	109	(8.2×)	49.5	74.5	51.4

Table 1. Results on CrowdPose *test* set [26]. Nano, Mobile, and Pi denote NVIDIA Jetson Nano GPU, Qualcomm Snapdragon 855, and Raspberry Pi 4B+, respectively. *LitePose* achieves better performance with up to 10.4× MACs reduction and 8.2× latency reduction on mobile platforms compared with HigherHRNet [8].

apply *once-for-all* [3] to automatically prune the redundancy in channels and select the optimal input resolution. The optimization goal and the search process are described in the following. Through NAS, we get four LitePose models (XS, S, M, and L) for different computation budgets. In Section 5.3, we show the effectiveness of NAS in detail.

Optimization Goal. Suppose that the original LitePose architecture contains $\{c_k\}_{k=1}^{K}$ channels in each layer, where K denotes the number of layers of the network. Our optimization goal is to find a sub-network whose input resolution is $r' < r$ with channel width $\{c'_k\}_{k=1}^{K}$ where $c'_k \leq c_k$, such that it could meet our efficiency constraint while achieving the best Average Precision (AP).

One-shot Supernet Training. We first train a LitePose supernet that supports different channel number configurations via weight sharing following [3, 12]. For each training iteration, we uniformly sample a channel configuration and train the supernet with it. In this way, each configuration is equally trained and could operate independently. To help the supernet learn better associate embedding [37] for grouping, we initialize the supernet with pre-trained weights. See Section 5.2 for more details about the supernet training and pre-training.

Search & Fine-tune. Since the supernet is thoroughly trained with weight sharing, we could directly extract the weights of a certain sub-network and evaluate the sub-network without further fine-tuning. This approximates the final performance of the sub-network. We use the evolutionary algorithm [42] to find the optimal configurations given specific efficiency constraints (*e.g.*, MACs). After finding optimal configurations, we fine-tune the corresponding sub-networks for several epochs and report the final performance. See Section 5 for more details about the fine-tuning.

5. Experiments

5.1. Dataset & Metrics

Microsoft COCO. Microsoft COCO [28] contains over 200,000 images with 250,000 person instances labeled with 17 keypoints. It is divided into *train/val/test-dev* sets with 57k, 5k, and 20k images, respectively. All our experiments on Microsoft COCO [28] are trained only on the *train* set. And we report the results on both *val* and *test-dev* sets.

CrowdPose. CrowdPose [26] consists of 20,000 images, containing about 80,000 persons labeled with 14 keypoints. Compared to Microsoft COCO [28], CrowdPose [26] contains more crowded scenes, posing more challenges to pose estimation methods. Following HigherHRNet [8], we train our models on the *train+val* set and report our results on the *test* set.

Evaluation Metric. The standard evaluation metric is based on Object Keypoint Similarity (OKS): $\text{OKS}=\frac{\sum_i exp(-d_i^2/2s^2k_i^2)\delta(v_i>0)}{\sum_i \delta(v_i>0)}$. Here d_i represents the Euclidean distance between a detected keypoint and its corresponding ground truth position. v_i denotes the visibility flag of keypoint i. s is the object scale, and k_i is a per-keypoint constant that controls falloff. Based on OKS, we report the standard average precision (AP), AP^{50}, and AP^{75} as the experiment results.

5.2. Experiment Setting

Data Augmentation. Following [8] and [51], the data augmentation includes random rotation $[-30°, 30°]$, random scale $[0.75, 1.5]$, random translation ($[-40, 40]$), and random flip.

Pre-training Details. We find that the network will learn low-quality Associative Embedding (AE) [37] if we train

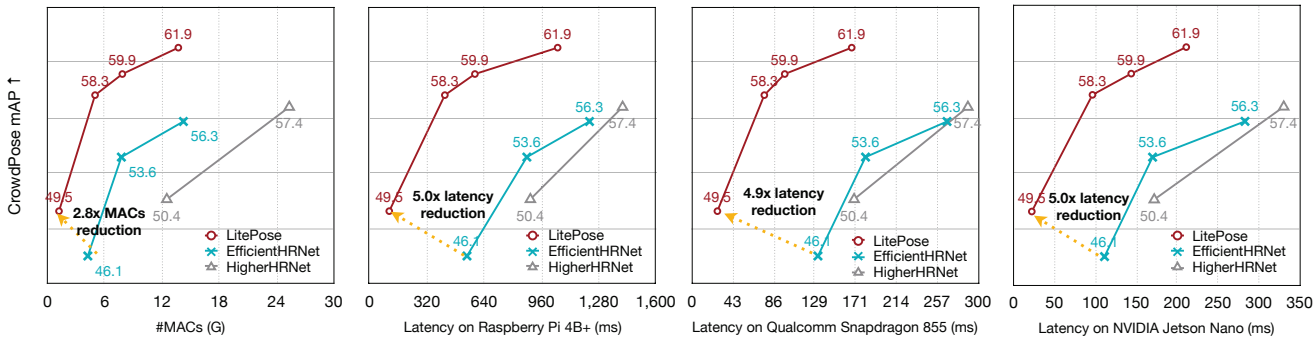


Figure 9. On CrowdPose [26], LitePose achieves 2.8× MACs reduction compared to EfficientHRNet [36]. The hardware-friendly design of LitePose allows high parallelism and therefore achieves much lower latency on various mobile platforms: It achieves 5.0×, 4.9×, and 5.0× latency reduction on Raspberry Pi 4B+, Qualcomm Snapdragon 855, and NVIDIA Jetson Nano respectively.

our one-shot supernet from scratch. To address this issue, we resort to pre-training. To be specific, we train the largest supernet without the AE loss (*i.e.*, only the heatmap loss) on the Microsoft COCO *train* set [28] for 100 epochs. Then we use it as the pre-trained model for further supernet training.

Supernet Training Setting. We conduct the one-shot NAS on the CrowdPose dataset [26]. We train *LitePose-L/M/S* and *LitePose-XS* with different training hyper-parameters and search space. We train *LitePose-L/M/S* supernet for 800 epochs with batch size 32 and *LitePose-XS* supernet for 2400 epochs with batch size 128. In each training step, we uniformly sample an architecture configuration from the search space and train the supernet with it ($lr = 0.001$ for $bs = 32$, $lr = 0.004$ for $bs = 128$).

Fine-tuning Setting. On CrowdPose dataset [26], we fix the architecture configuration and tune the model for 200 epochs with batch size 32. The original learning rate is set to 10^{-3}, and drops to 10^{-4} and 10^{-5} at the 50_{th} and the 180_{th} epoch, respectively (linearly increase [16] for $bs = 128$ case). On COCO dataset [28], we take the supernet trained on CrowdPose [26] as the pre-trained model for initialization. For each searched configuration, we train the corresponding model for 500 epochs with batch size 32. The original learning rate is set to 10^{-3}, dropped to 10^{-4} and 10^{-5} at the 350_{th} and the 480_{th} epoch, respectively.

Search Details. We conduct NAS on the CrowdPose dataset [26]. After obtaining the searched architectures, we directly generalize them to the COCO dataset [28] and report their performance on both datasets. For *LitePose-L/M/S* supernet training, we choose resolution from $[512, 448]$ and channel width ratio from $[1.0, 0.75, 0.5]$. For *LitePose-XS* supernet training, we choose resolution from $[512, 448, 384, 320, 256]$ and channel width ratio from $[1.0, 0.75, 0.5, 0.25]$.

Measurement Details. We measure the latency of our models on Qualcomm Snapdragon 855 GPU, Raspberry Pi 4B+, and NVIDIA Jetson Nano GPU. For real-world edge deployment, it is crucial for DL models to efficiently integrate some optimized libraries and runtimes as their backends and generate the fastest possible executable. Therefore, all the latency results we report on raspberry Pi 4B+ and NVIDIA Jetson Nano GPU are optimized by TVM AutoScheduler [6, 56], which can help us better simulate the latency of real-world applications.

5.3. Ablation Experiments

Large Kernels. As shown in Table 3 and Figure 7, with only minor computation increase, the 7×7 kernels enhance the capability of coping with scale variation problem and therefore provides the best performance.

Fusion Deconv Head. Another way to handle the scale variation problem is multi-resolution fusion as the introduction of large resolution features can help better capture small persons. We quantitatively show the performance gain in Table 3 and Figure 6: our efficient fusion deconv head improve the performance by +7.6AP on CrowdPose [26] dataset with only minor computation increase.

Neural Architecture Search. Neural Architecture Search (NAS) benefits our method from two aspects: one-shot supernet training and architecture search with fine-tuning. As shown in Table 3, supernet training provides +1.4AP and +2.7AP on *0.5 LitePose* and *LitePose-XS*, respectively. Architecture search also offers +2.2AP on *0.5 LitePose*. Besides, for *LitePose-XS*, we use *LitePose-S* as its teacher for heatmap loss in fine-tuning and obtain +1.1AP.

5.4. Main Results

Results on CrowdPose. We first report the results on the CrowdPose dataset [26]. Compared to Microsoft COCO [28],

Model	Input Size	#Params	#MACs		Latency (ms)						AP_{val}	$AP_{test\text{-}dev}$
					Nano		Mobile		Pi			
PersonLab [40]	1401×1401	68.7M	405.5G	–	–		–		–		66.5	66.5
Hourglass [38]	512×512	277.8M	206.9G	–	–		–		–		–	56.6
HigherHRNet-W48 [8]	640×640	63.8M	155.1G	–	2101	–	1532	–	12302	–	69.9	68.4
Lightweight OpenPose [39]	368×368	4.1M	9.0G	–	155	–	97	–	562	–	42.8	–
EfficientHRNet-H_{-2} [36]	448×448	8.3M	7.9G	(1.1×)	170	(0.9×)	182	(0.5×)	878	(0.6×)	52.9	52.8
LitePose-S (Ours)	448×448	2.7M	5.0G	(1.8×)	97	(1.3×)	76	(1.3×)	420	(1.3×)	56.8	56.7
EfficientHRNet-H_{-4}	384×384	2.8M	2.2G	–	50	–	78	–	273	–	35.7	35.5
LitePose-XS (Ours)	256×256	1.7M	1.2G	(1.8×)	22	(2.3×)	27	(2.9×)	109	(2.5×)	40.6	37.8

Table 2. Results on COCO *val/test-dev* set [28]. Compared with EfficientHRNet [36], LitePose achieves 1.8× MACs reduction and up to 2.9× latency reduction while providing better performances. Compared with Lightweight OpenPose [39], it obtains much higher performance (+14.0AP) with lower latency.

Arch	Setting				#MACs	AP
	Knl.	Fsn.	Spnt.	Dstl.		
0.5 LitePose	3 × 3	✓			3.8G	41.7
0.5 LitePose	5 × 5	✓			4.1G	50.7
0.5 LitePose	7 × 7				4.1G	47.1
0.5 LitePose	7 × 7	✓			4.7G	54.7
0.5 LitePose	7 × 7	✓	✓		4.7G	56.1
LitePose-S (Ours)	7 × 7	✓	✓		5.0G	58.3
LitePose-XS	7 × 7	✓			1.2G	45.7
LitePose-XS	7 × 7	✓	✓		1.2G	48.4
LitePose-XS (Ours)	7 × 7	✓	✓	✓	1.2G	49.5

Table 3. Ablation study. **0.5 LitePose** means linearly scales each layer to the LitePose supernet to 50% channels. **Knl.**: Kernel size; **Fsn**: Fusion deconv head; **Spnt.**: Supernet training; **Dstl**: Distillation. Large kernels and the fusion deconv head provide +13.0AP and +7.6AP, respectively. The supernet training benefits both two configurations by up to +2.7AP. The architecture search brings +2.2AP to the manually designed model. And the distillation brings +1.1AP to *LitePose-XS*.

top-down设计在某些场景下会缺点

it contains more crowded scenes. The strong assumption of top-down methods that each person detection box only contains a single person in the center is hard to satisfy in crowded scenes. Therefore, several top-down methods [10, 15] that perform well on the COCO dataset [28] fail on the CrowdPose dataset [26]. We also achieve a much better performance-computation trade-off than the state-of-the-art bottom-up HRNet-based baselines [8, 36]. As shown in Table 1 and Figure 9, our architecture achieves 2.8× MACs reduction and up to 5.0× latency reduction on mobile platforms compared with HRNet-based methods.

Results on Microsoft COCO. We also show the results on Microsoft COCO dataset [28]. Our method outperforms HRNet-based approaches [8, 36] by a large margin. As shown in Table 2 and Figure 1, our architecture achieves up to 2.4× with even higher performance. Besides, compared with Lightweight Openpose [39], our method performs much better (+14.0AP) with lower latency on mobile platforms.

6. Conclusion

In this paper, we studied efficient architecture design for multi-person pose estimation on edge. We designed a gradual shrinking experiment to bridge the multi-branch and single-branch architecture. Our study shows that the high-resolution branches are redundant for models in the low-computation region. Inspired by this, we propose LitePose, an efficient architecture for pose estimation, which inherits the merits of both the single-branch and multi-branch architecture. Extensive experiments demonstrate the effectiveness and robustness of LitePose, paving the way to real-time human pose estimation for edge applications.

Acknowledgement

We thank National Science Foundation, MIT-IBM Watson AI Lab, Ford, Hyundai and Intel for supporting this research. We thank Yaoyao Ding and Lianmin Zheng for their helpful comments on TVM. We also thank Shengyu Wang and Ruihan Gao for their valuable feedback on the manuscript.

Limitations and Future Work

Though we take a big step towards real-time human pose estimation, the computational cost (1 GMACs) is still too large for more extremely resource-limited edge devices (*e.g.*, micro-controller). Also, the depth-wise convolutions are not well supported on existing frameworks (*e.g.*, PyTorch, TensorFlow). Besides, our *LitePose* cannot achieve its best performance without the help of specific inference backends (*e.g.*, TVM AutoScheduler [6, 56]).

References

[1] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559. PMLR, 2018.

[2] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[5] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[6] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 578–594, 2018.

[7] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7103–7112, 2018.

[8] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5386–5395, 2020.

[9] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[10] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2334–2343, 2017.

[11] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression. *arXiv preprint arXiv:2104.02300*, 2021.

[12] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020.

[13] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[14] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[16] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[17] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[18] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[21] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.

[22] Sheng Jin, Wentao Liu, Enze Xie, Wenhai Wang, Chen Qian, Wanli Ouyang, and Ping Luo. Differentiable hierarchical graph grouping for multi-person pose estimation. In *European Conference on Computer Vision*, pages 718–734. Springer, 2020.

[23] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[24] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11977–11986, 2019.

[25] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[26] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10863–10872, 2019.

[27] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2178–2188, 2017.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[29] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018.

[30] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

[31] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[32] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[33] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3296–3305, 2019.

[34] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *arXiv preprint arXiv:1907.03739*, 2019.

[35] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[36] Christopher Neff, Aneri Sheth, Steven Furgurson, and Hamed Tabkhi. Efficienthrnet: Efficient scaling for lightweight high-resolution multi-person pose estimation. *arXiv preprint arXiv:2007.08090*, 2020.

[37] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *arXiv preprint arXiv:1611.05424*, 2016.

[38] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[39] Daniil Osokin. Real-time 2d multi-person pose estimation on cpu: Lightweight openpose. *arXiv preprint arXiv:1811.12004*, 2018.

[40] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286, 2018.

[41] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4929–4937, 2016.

[42] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[44] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[45] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[46] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.

[47] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[48] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665*, 2016.

[49] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[50] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

[51] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018.

[52] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[53] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A lightweight high-resolution network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10440–10450, 2021.

[54] Wenqiang Zhang, Jiemin Fang, Xinggang Wang, and Wenyu Liu. Efficientpose: Efficient human pose estimation with neural architecture search. *arXiv preprint arXiv:2012.07086*, 2020.

[55] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[56] Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Wu, Cody Hao Yu, Ameer Haj-Ali, Yida Wang, Jun Yang, Danyang Zhuo, Koushik Sen, et al. Ansor: Generating high-performance tensor programs for deep learning. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, pages 863–879, 2020.

[57] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[58] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.