EfficientHRNet and Lite-HRNet

Marco Sutti

June 28, 2021

I. EfficientHRNet

Overview

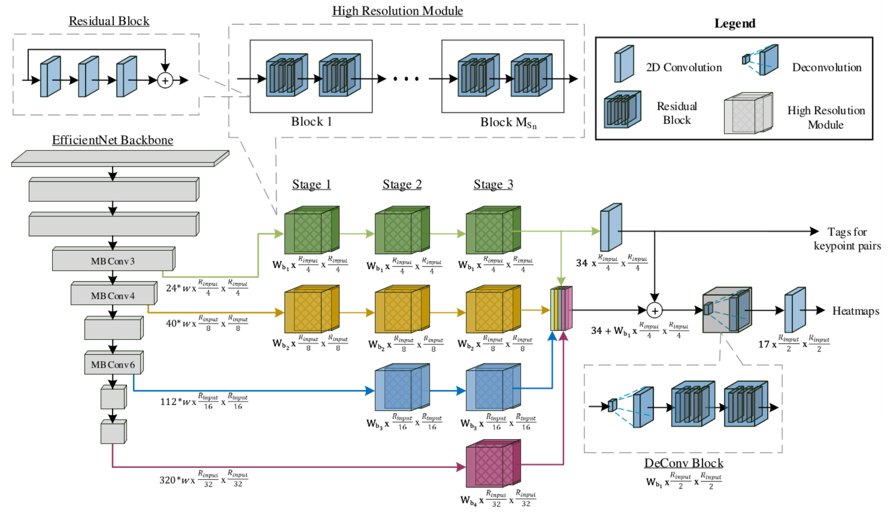
- Real-time multi-person **2D pose estimation**.
- **Scalable** algorithm.
- **Highly accurate** models while **reducing computation**.
- **Lightweight bottom-up method** $\rightsquigarrow$ execution under constrained computational resources (e.g., IoT devices).

EfficientHRNet: [Neff et al. '20]

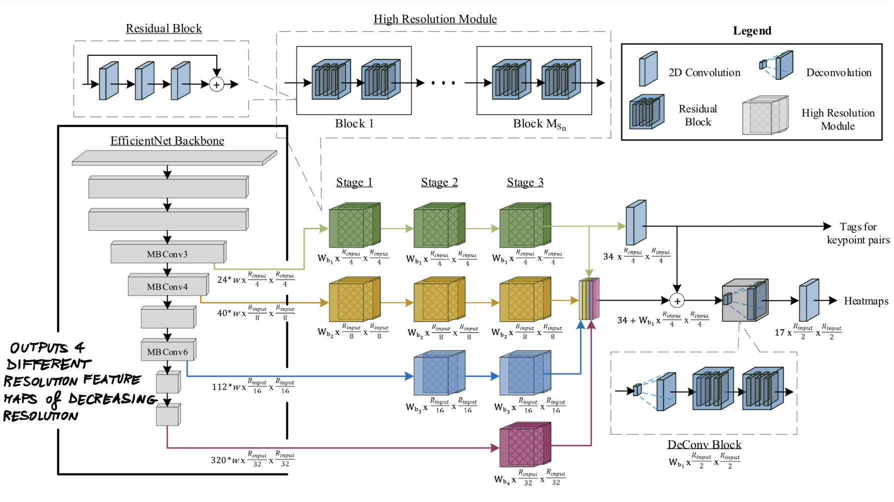
EfficientHRNet – Main ideas

- **EfficientHRNet** unifies **EfficientNet + HRNet** principles.
  - Like **HRNet**, it uses **multiple resolutions** of features
  - Uses **EfficientNet** as a **backbone** and adapts its **scaling methodology** $\rightsquigarrow$ **Scale below the baseline** resolution B0 + **Jointly scale down** the input resolutions, High-Resolution Network, and Heatmap Prediction Network.
- **Compound scaling** inspired by EfficientNet, jointly scales the width, depth and input resolution of EfficientHRNet.
- This leads to a **family of lightweight and scalable networks** flexible towards **accuracy** and **computation requirements**.

Network architecture

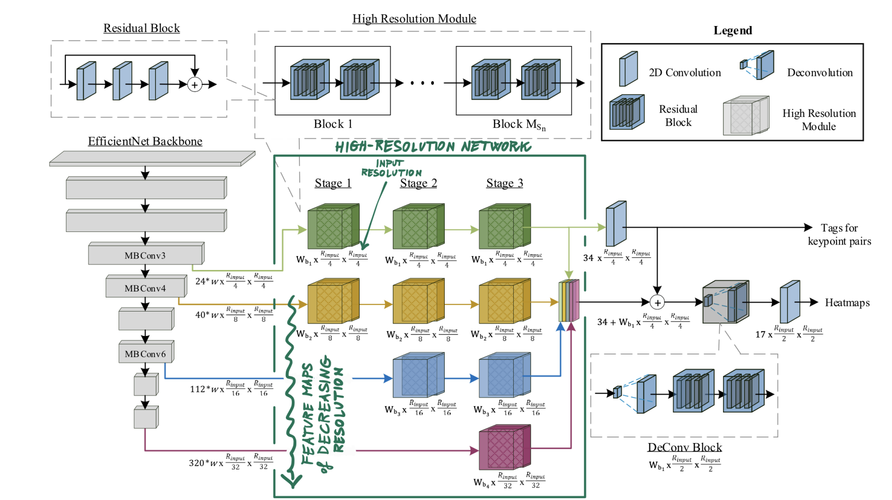


Backbone Network



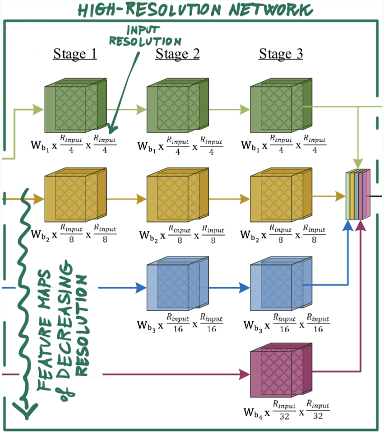
- **First stage: Backbone**, it is a modified EfficientNet.

High-Resolution Network



- **Main body**: **High-Resolution Network**.

High-Resolution Network



- It has **three stages** s_1, s_2, and s_3, containing **four parallel branches** b_1, b_2, b_3, and b_4 of different resolutions.
- The first stage starts with 2 branches, with each consecutive stage adding an additional branch.
- Each branch b_n consists of ***high resolution modules*** with a width of W_{b_n}, and contains feature representations of decreasing resolutions

$$W_{b_n} \times \frac{R_{input}}{2^{n+1}},$$

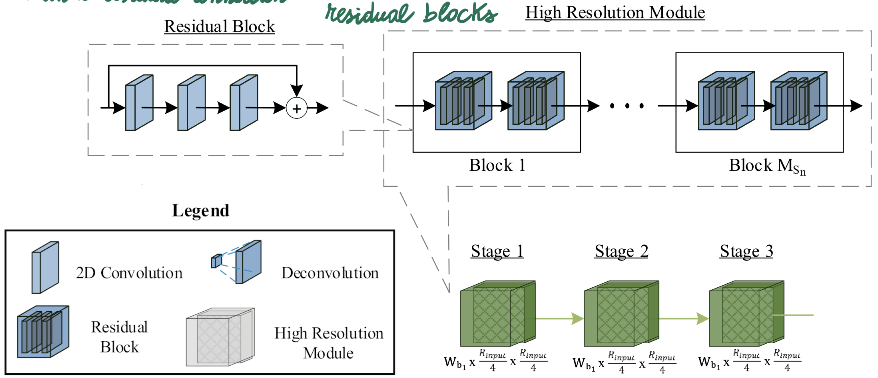
where R_{input} is the original input resolution.

High-Resolution Network

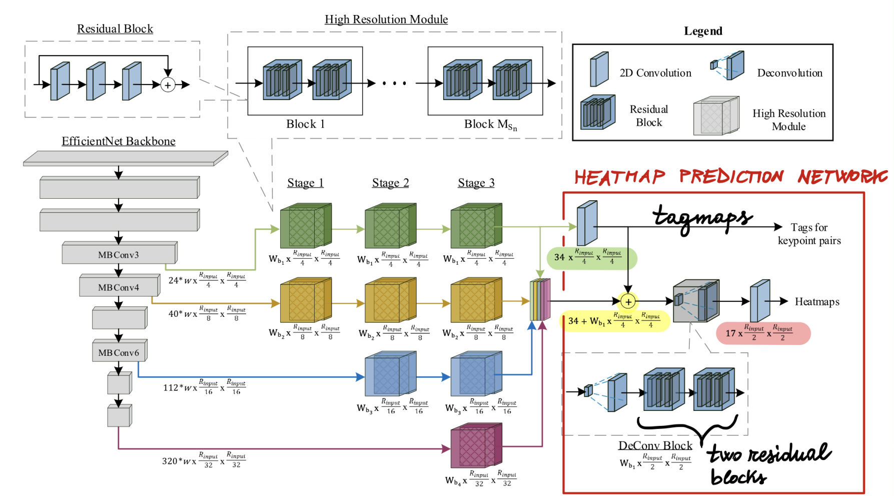
1 Each high resolution module is made up of M_{sn} blocks

2 Each block contains two residual blocks

3 Each residual block performs 3 convolutions with a residual connection

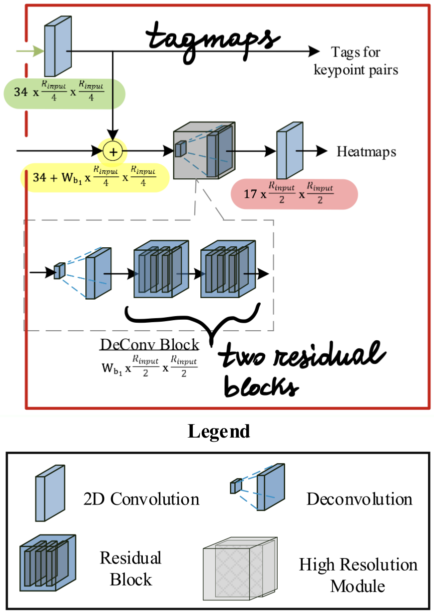


Heatmap Prediction Network



- **Heatmap Prediction Network**: used to generate **human keypoint predictions**.

Heatmap Prediction Network



- To predict more accurate heatmaps, a ***DeConv block*** is added.
- Input to the ***DeConv block***: **concatenation** of features maps and predicted heatmaps from the **High-Resolution Network**.
- Two ***residual blocks*** are added after the deconvolution to refine the up-sampled feature maps.
- 1x1 convolution is used to predict tagmaps and heatmaps:

$$T_{size} = 34 \times \frac{R_{input}}{4} \times \frac{R_{input}}{4},$$

$$H_{size} = 17 \times \frac{R_{input}}{2} \times \frac{R_{input}}{2}.$$

- **Heatmaps loss**: sum of MSEs for all resolutions.

Compound Scaling

- **Jointly scales all part of EfficientHRNet** to meet a diverse set of memory and compute constraints.
- **Heuristic-based** compound scaling methodology.
- **EfficientHRNet** uses a scaling coefficient ϕ to **jointly** scale:
  - the **Backbone Network**,
  - the **High-Resolution Network**,
  - the Task-Specific Head.

Compound Scaling/Backbone Network

- The **EfficientNet backbone** is scaled below the baseline.
- Starting with the baseline **EfficientNet-B0** scaling coefficients:
  - ***depth***: $d = 1.2^{\phi}$,
  - ***width***: $w = 1.1^{\phi}$,
  - ***resolution***: $r = 1.15^{\phi}$,

  $\phi = -1, -2, -3, -4$ is used to calculate the scaling multipliers for the **compact EfficientNet** models B_{-1}, B_{-2}, B_{-3}, and B_{-4}.
- **Example**:

  To scale the baseline resolution 224 down for our B_{-1} model, we take r with $\phi = -1$:

$$\text{ceil}(224 \cdot 1.15^{-1}) = 195.$$

Model	Input size
B0	224
B_{-1}	195
B_{-2}	170
B_{-3}	145
B_{-4}	128

Compound Scaling/High-Resolution Network

- The baseline H_0 has width of 32, 64, 128, 256 for each branch n, respectively.
- Scale them down with a ***width*** scaling factor of 1.25:

$$W_{b_n} = (n \cdot 32) \cdot (1.25)^{\phi}.$$

  - **Example**: For $n = 1$, $\phi = -1$, we get

$$W_{b_n} = (1 \cdot 32) \cdot (1.25)^{-1} = 32/1.25 \approx 26.$$

- The input resolution of EfficientHRNet is linearly scaled down:

$$R_{input} = 512 + 32 \cdot \phi.$$

Model	Input Size (R_{input})	Backbone Network	Width per Branch (W_{b_1}, W_{b_2}, W_{b_3}, W_{b_4})	Blocks per Stage (M_{s_2}, M_{s_3}, M_{s_4})	Tags (T_{size})	Heatmaps (H_{size})
H_0 ($\phi = 0$)	512	$B0$	32, 64, 128, 256	1, 4, 3	128	256
H_{-1} ($\phi =$ -1)	480	B_{-1}	26, 52, 103, 206	1, 3, 3	120	240
H_{-2} ($\phi =$ -2)	448	B_{-2}	21, 42, 83, 166	1, 2, 3	112	224
H_{-3} ($\phi =$ -3)	416	B_{-3}	17, 34, 67, 133	1, 1, 3	104	208
H_{-4} ($\phi =$ -4)	384	B_{-4}	14, 27, 54, 107	1, 1, 2	96	192

Experiments/2D Human Pose Estimation

Comparison of EfficientHRNet with other bottom-up pose estimation methods on COCO2017 *test-dev* set (a subset of *test* with 20k images used for fair comparison with other works).

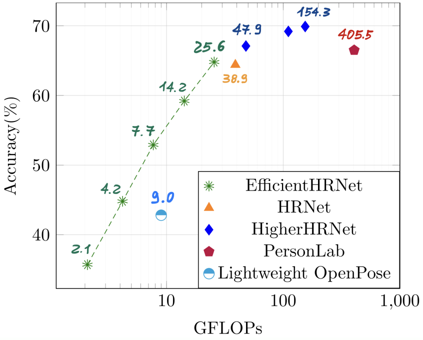
Method	Backbone	Input size	# Params	FLOPs	AP
w/o multi-scale test					
OpenPose	-	-	25.94M	160B	61.8
Hourglass	Hourglass	512	277.8M	206.9B	56.6
PersonLab	ResNet-152	1401	68.7M	405.5B	66.5
PifPaf	ResNet-152	-	-	-	66.7
HRNet	HRNet-W32	512	28.5M	38.9B	64.1
HigherHRNet	HRNet-W32	512	28.6M	47.9B	66.4
HigherHRNet	HRNet-W48	640	63.8M	154.3B	68.4
H_0	$B0$	512	23.3M	25.6B	64.0
H_{-1}	B_{-1}	480	16M	14.2B	59.1
H_{-2}	B_{-2}	448	10.3M	7.7B	52.8
H_{-3}	B_{-3}	416	6.9M	4.2B	44.5
H_{-4}	B_{-4}	384	3.7M	2.1B	35.5
w/ multi-scale test					
Hourglass	Hourglass	512	277.8M	206.9B	63.0
Hourglass	Hourglass	512	277.8M	206.9B	65.5
PersonLab	ResNet-152	1401	68.7M	405.5B	68.7
HigherHRNet	HRNet-W48	640	63.8M	154.3B	70.5
H_0	$B0$	512	23.3M	25.6B	67.1
H_{-1}	B_{-1}	480	16M	14.2B	62.3
H_{-2}	B_{-2}	448	10.3M	7.7B	55.0
H_{-3}	B_{-3}	416	6.9M	4.2B	45.5
H_{-4}	B_{-4}	384	3.7M	2.1B	39.7

- The **baseline H_0 model** w/o multi-scale test vs **HRNet**: **only 0.1% decrease** in **accuracy**, but $1 - 23.3/28.5 \approx$ **18% reduction in parameters** and $1 - 25.6/38.9 \approx$ **34%** in **FLOPs**!
- The H_{-1} model outperforms both **OpenPose** and **Hourglass**.
- As EfficientHRNet is scaled down, we see **minor drops in accuracy with significant drops in parameters and FLOPs** as compared to H_0.
- The lightest model H_{-4} w.r.t. H_0 is $1 - 3.7/23.3 \approx$ **84%** smaller and has $1 - 2.1/25.6 \approx$ **91.7%** less FLOPs, and **27.4%** drop in accuracy.

Experiments/2D Human Pose Estimation

Comparison of EfficientHRNet with other bottom-up pose estimation methods on COCO2017 *test-dev* set (a subset of *test* with 20k images used for fair comparison with other works).

Method	Backbone	Input size	# Params	FLOPs	AP
w/o multi-scale test					
OpenPose	-	-	25.94M	160B	61.8
Hourglass	Hourglass	512	277.8M	206.9B	56.6
PersonLab	ResNet-152	1401	68.7M	405.5B	66.5
PifPaf	ResNet-152	-	-	-	66.7
HRNet	HRNet-W32	512	28.5M	38.9B	64.1
HigherHRNet	HRNet-W32	512	28.6M	47.9B	66.4
HigherHRNet	HRNet-W48	640	63.8M	154.3B	68.4
H_0	$B0$	512	23.3M	25.6B	64.0
H_{-1}	B_{-1}	480	16M	14.2B	59.1
H_{-2}	B_{-2}	448	10.3M	7.7B	52.8
H_{-3}	B_{-3}	416	6.9M	4.2B	44.5
H_{-4}	B_{-4}	384	3.7M	2.1B	35.5
w/ multi-scale test					
Hourglass	Hourglass	512	277.8M	206.9B	63.0
Hourglass	Hourglass	512	277.8M	206.9B	65.5
PersonLab	ResNet-152	1401	68.7M	405.5B	68.7
HigherHRNet	HRNet-W48	640	63.8M	154.3B	70.5
H_0	$B0$	512	23.3M	25.6B	67.1
H_{-1}	B_{-1}	480	16M	14.2B	62.3
H_{-2}	B_{-2}	448	10.3M	7.7B	55.0
H_{-3}	B_{-3}	416	6.9M	4.2B	45.5
H_{-4}	B_{-4}	384	3.7M	2.1B	39.7



Experiments/Real-Time Execution on IoT Edge Devices

Compare accuracy and efficiency across differing platforms.

- **Accuracy · Efficiency (Æ) metric**: product of accuracy (measured in AP) and efficiency (measured in FPS per Watt).

Model	AP	FPS	Efficiency	Æ
HigherHRNet	67.1	6.68	0.445	29.850
Lightweight OpenPose	42.8	26	0.578	24.738
H_0 ($\phi = 0$)	64.8	22.95	1.530	99.144
H_{-1} ($\phi =$ -1)	59.2	20.43	1.362	80.630
H_{-2} ($\phi =$ -2)	52.9	24.53	1.635	86.492
H_{-3} ($\phi =$ -3)	44.8	33.78	2.252	100.89
H_{-4} ($\phi =$ -4)	35.7	50.96	3.397	121.273

- EfficientHRNet outperforms the competition between 3x to 5x.
- HigherHRNet excels in accuracy and Lightweight OpenPose excels in FPS and model size, while EfficientHRNet is more equally balanced between accuracy, model size, throughput, and power consumption.
- This makes **EfficientHRNet** the **state-of-the-art method** for **lightweight bottom-up human pose estimation for real-time edge applications**.

II. Lite-HRNet

Lite-HRNet

- First, combine the **shuffle block** in ShuffleNet and the **high-resolution design pattern** in HRNet.

  $\rightsquigarrow$ **naive Lite-HRNet.**

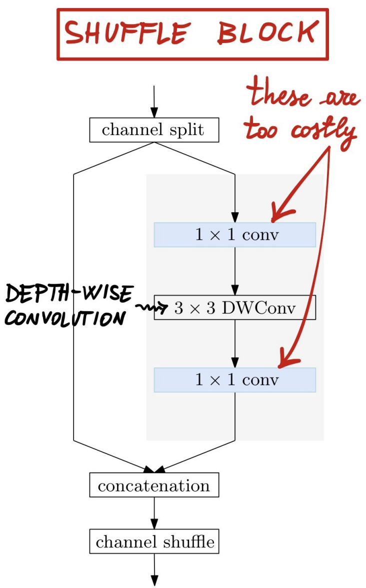
- Second, introduce an efficient **conditional channel weighting** to replace the costly pointwise (1×1) convolutions in shuffle blocks.

  $\rightsquigarrow$ **Lite-HRNet.**

  - It computes the weights from all the channels and uses them as a **bridge to exchange information across channels and resolutions**.
  - **Reduction in computational complexity!**

Lite-HRNet: [Yu et al. '21]

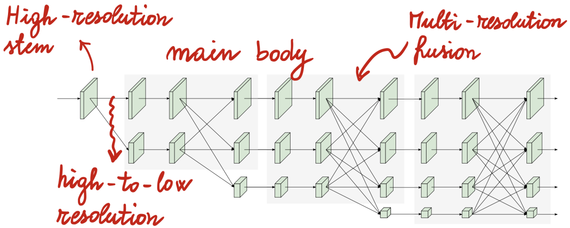
Naive Lite-HRNet/Shuffle Blocks



ShuffleNet: [Ma et al. '18]

- The shuffle block in ShuffleNet V2 first splits the channels into **two partitions**.
  - One partition passes through a sequence of 1×1 convolution, 3×3 depthwise convolution, and 1×1 convolution.
  - The output is **concatenated** with the other partition.
- Finally, the concatenated channels are **shuffled**.

Naive Lite-HRNet/HRNet

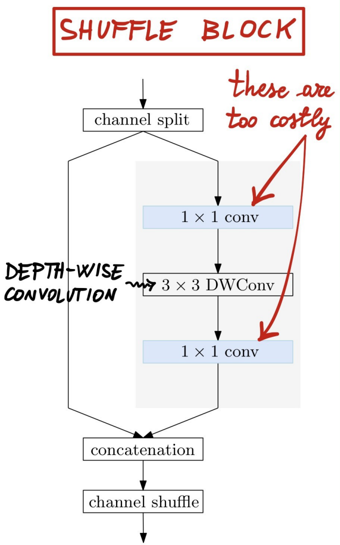


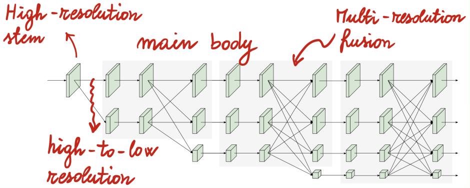
- **First stage**: high-resolution convolution stem. Consists of two 3×3 convolutions.
- Adds **high-to-low resolution streams** one by one as new stages.
  - The multi-resolutions streams are connected in parallel.
  - The output is concatenated with the other partition.
- **Main body**: sequence of stages. In each stage:
  - the **information across resolutions** is **exchanged** repeatedly;
  - a **sequence of residual blocks** and **one multi-resolution fusion**.

HRNet: [Wang et al. '19]

Naive Lite-HRNet/Simple Combination

⇝ **Shuffle blocks + HRNet = Naive Lite-HRNet**





- Use the **shuffle block** to replace:
  - the second 3 × 3 convolution in the stem of HRNet;
  - all the normal residual blocks (formed with two 3 × 3 convolutions).

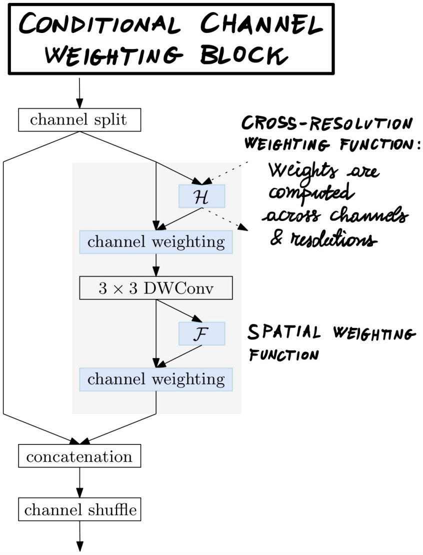
Lite-HRNet

- **Motivation**: 1×1 **convolution is costly**, since it performs a matrix-vector multiplication at each position:

$$\mathsf{Y} = \mathbf{W} \otimes \mathsf{X},$$

  where X and Y are input and output maps, and $\mathbf{W}$ is the 1×1 **convolutional kernel**.
- It serves the critical role of **exchanging information across channels**.
- It has **quadratic time complexity** $\Theta(C^2)$ w.r.t. C, the number of channels.

Lite-HRNet/Conditional Channel Weighting (CCW)



- Use the **element-wise weighting operation** to replace the 1×1 convolution in naive Lite-HRNet.
- **Element-wise weighting operation** for the sth resolution branch:

$$\mathsf{Y}_s = \mathsf{W}_s \odot \mathsf{X}_s,$$

  where $\mathsf{W}_s \in \mathbb{R}^{W_s \times H_s \times C_s}$ is a **weight map**, and $\mathsf{X}_s \in \mathbb{R}^{H_s \times W_s \times C_s}$ is the input channel map for the s resolution.
- It has **linear complexity** $\Theta(C)$.
- We compute the weights by using the channels for a single resolution and the channels across all the resolutions.
- Weights play a role of **exchanging information across channels and resolutions**.

Experiments

Complexity and accuracy comparison of Lite-HRNet on the COCO and MPII datasets.

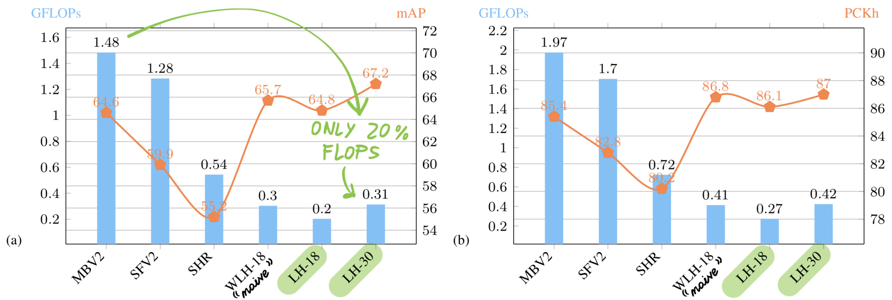


Figure 4. **Illustration of the complexity and accuracy comparison on the COCO `val` and MPII `val` sets.** (a) Comparison on COCO `val` with 256 × 192 input size. (b) Comparison on MPII `val` with 256 × 256 input size. MBV2= MobileNet V2. SFV2= ShuffleNet V2. SHR= Small HRNet-W16. (W)LH= (Wider Naive) Lite-HRNet.

- Compared to MobileNetV2, Lite-HRNet-30 improves AP by 2.6 points, **with only** 20% **GFLOPs and parameters!**
- Compared to ShuffleNetV2, Lite-HRNet-18 and Lite-HRNet-30 improve AP by 4.9 and 7.3 points, **with only** 15% **and** 24% **GFLOPs and parameters!**

Conclusion

This talk:

- **EfficientHRNet** and **Lite-HRNet**: **lightweight** and **high resolution networks**, for human body pose estimation.
- Both can be deployed on **resource-constrained**, **very low power devices**.
- High degree of flexibility due to their **scalable structure**.

Thank you for listening!

III. Bonus material

Experiments/Classification for EfficientNet

Compact EfficientNet performance on ImageNet and CIFAR-100 datasets.

- Looking at B_{-1} there are:
  - a $1 - 4.5/5.3 \approx$ **15% reduction in parameters**;
  - a $1 - 0.3/0.4 =$ **25% reduction in operations**;
  - yet an **accuracy drop** of only **1.2%** and **0.5%** on ImageNet and CIFAR-100, respectively.
- In the most extreme, B_{-4} shows:
  - a $1 - 1.3/5.3 \approx$ **75% reduction in parameters**;
  - a $1 - 0.05/0.4 =$ **87.5% reduction in operations**;
  - with an **accuracy drop** of **9.4%** and **7.6%** on ImageNet and CIFAR-100, respectively.

$\rightsquigarrow$ **The massive reduction in computation allows for great flexibility.**

Model	Input size	FLOPs	ImageNet Params	ImageNet Top-1	CIFAR-100 Params	CIFAR-100 Top-1
B0	224	0.4B	5.3M	75	4.1M	81.9
B_{-1}	195	0.3B	4.5M	73.8	3.5M	81.4
B_{-2}	170	0.2B	3.4M	71.3	2.5M	79.8
B_{-3}	145	0.1B	2.8M	68.5	1.9M	78.2
B_{-4}	128	0.05B	1.3M	65.6	1.3M	74.3

Lite-HRNet/Structure of Lite-HRNet

Table 1. **Structure of Lite-HRNet**. The stem contains one stride 2 3×3 convolution and one shuffle block. The main body has three stages, each of which has a sequence of modules. Each module consists of two conditional channel weight blocks and one fusion block. N in Lite-HRNet-N indicates the number of layers. *resolution branch* indicates this stage contains the feature stream of the corresponding resolution. ccw = conditional channel weight.

layer	output size	operator	resolution branch	*#output_channels*	repeat	*#modules*	
						Lite-HRNet-18	Lite-HRNet-30
image	256×256		$1\times$	3			
stem	64×64	conv2d	$2\times$	32	1	1	1
		shuffle block	$4\times$	32	1		
stage_2	64×64	ccw block	$4\times$ $8\times$	40, 80	2	2	3
		fusion block	$4\times$ $8\times$	40, 80	1		
stage_3	64×64	ccw block	$4\times$ $8\times$ $16\times$	40, 80, 160	2	4	8
		fusion block	$4\times$ $8\times$ $16\times$	40, 80, 160	1		
stage_4	64×64	ccw block	$4\times$ $8\times$ $16\times$ $32\times$	40, 80, 160, 320	2	2	3
		fusion block	$4\times$ $8\times$ $16\times$ $32\times$	40, 80, 160, 320	1		
FLOPs						273.4M	425.3M
#Params						1.1M	1.8M

Lite-HRNet/Cross-Resolution Weight Computation

Example of complexity comparison between 1×1 convolutions and depthwise convolutions.

Table 2. **Computational complexity comparison:** 1×1 **convolution vs. conditional channel weight**. $\mathsf{X}_s \in \mathcal{R}^{H_s \times W_s \times C_s}$ are the input channel maps for the s resolution, X_1 corresponds to the highest resolution. $N_s = H_s W_s$. For example, the shape of X_1 and X_2 are $64 \times 64 \times 40$ and $32 \times 32 \times 80$, respectively. single/cross-resolution=single/cross resolution information exchange.

model	single-resolution	cross-resolution	Theory Complexity	Example FLOPs
1×1 convolution	✓		$\sum_1^s N_s C_s^2$	12.5M
3×3 depthwise convolution			$\sum_1^s 9 N_s C_s$	2.1M
CCW w/ spatial weights	✓		$\sum_1^s (2C_s^2 + N_s C_s)$	0.25M
CCW w/ multi-resolution weights		✓	$2(\sum_1^s C_s)^2 + \sum_1^s N_s C_s$	0.26M
CCW	✓	✓	$2(\sum_1^s C_s)^2 + 2\sum_1^s (C_s^2 + N_s C_s)$	0.51M

Lite-HRNet/Cross-Resolution Weight Computation/1

- Considering the s stage, there are s parallel resolutions, and s **weight maps** $\mathsf{W}_1, \mathsf{W}_2, \ldots, \mathsf{W}_s$.
- We compute the s weight maps from all the channels across resolutions using a **lightweight** function $\mathcal{H}_s$:

$$(\mathsf{W}_1, \mathsf{W}_2, \ldots, \mathsf{W}_s) = \mathcal{H}_s(\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_s),$$

  where $\{\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_s\}$ are the input maps for the s resolutions. $\mathsf{X}_1 \rightsquigarrow$ highest resolution, $\mathsf{X}_2 \rightsquigarrow$ sth highest resolution.

Lite-HRNet/Cross-Resolution Weight Computation/2

Implementation of the lightweight function $\mathcal{H}_s$:

- Perform **adaptive average pooling (AAP)** on $\{\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_{s-1}\}$:

$$\mathsf{X}_1' = \mathrm{AAP}(X_1), \quad \mathsf{X}_2' = \mathrm{AAP}(X_2), \quad \ldots, \quad \mathsf{X}_{s-1}' = \mathrm{AAP}(X_{s-1}).$$

  $\rightsquigarrow$ AAP pools any input size to a given output size $W_s \times H_s$.

- **Concatenate** $\{\mathsf{X}_1', \mathsf{X}_2', \ldots, \mathsf{X}_{s-1}'\}$ and X_s together, followed by a 1×1 convolution, ReLU, 1×1 convolution, and sigmoid, generating weight maps consisting of s partitions, $\mathsf{W}_1', \mathsf{W}_2', \ldots, \mathsf{W}_s$:

$$(\mathsf{X}_1', \mathsf{X}_2', \ldots, \mathsf{X}_s) \rightarrow \text{Conv.} \rightarrow \text{ReLU} \rightarrow \text{Conv.} \rightarrow \sigma \rightarrow (\mathsf{W}_1', \mathsf{W}_2', \ldots, \mathsf{W}_s).$$

- The $s-1$ weight maps $\mathsf{W}_1', \mathsf{W}_2', \ldots, \mathsf{W}_{s-1}'$ are **upsampled** to the corresponding resolutions, outputting $\mathsf{W}_1, \mathsf{W}_2, \ldots, \mathsf{W}_{s-1}$ for the subsequent element-wise channel weighting.