Searching for Neural Fabric for Body Part Representation with Vector in Pixel

Sen Yang*

Abstract

For human pose estimation, deep, multi-scale and stackable neural architecture design can improve the prediction accuracy, but is limited by computational budgets. In this paper, we propose a compact cell-based neural fabric which can be applied with neural architecture search method. Not satisfied with achieving a lightweight neural architecture, we also take full advantage of the flexibility of our design to explore a new representation for human pose. The articulated pose constraint relationship is divided into multiple part representations respectively learned by multiple subnetworks as an alternative to learning the global implicit relationship of pose. In addition, a novel *vector in pixel* approach is introduced to part representation, by which we use the ℓ_2 norm of vector to represent the existing score of keypoints in pixel position and its orientation capture more local component information of body part to alleviate feature ambiguity caused by strong supervision. We demonstrate the effectiveness of our method on two benchmarks datasets MPII and MS-COCO, achieving competitive results compared with state-of-the-art performance.

1 Introduction

Given a specified computer vision task, architecture design and the representation of output are two key issues to be considered for methods based on deep convolutional neural network. For image classification, fundamental chain-like convolution neural networks can map a RGB image to a fixed-length vector representing categories' probability. Semantic segmentation, instance segmentation and pose estimation require network's output to encode location and semantics. Accordingly, heatmap representation can replace vector as the output of network and fully convolutional network (FCN) and more sophisticated networks are designed to make prediction at every pixel. In this work, we focus on these two aspects to handle human pose estimation task as well.

Automated neural architecture search (NAS) play a potential role at designing neural architecture for various tasks as an alternative to human expert. However, the general form of architecture search space is defined by human as usual. One-shot model design paradigm for NAS aims to search for a micro repeatable cell or motif sharing weights, yet macro design of meta-architecture upon cells still needs important consideration.

Borrowing homogeneous local connection pattern from convolutional neural fabric, we "weave" cells into a neural fabric to enable the whole architecture compact internally and externally. Motivated by DARTS, we relax discrete architecture into continuous space to make it searchable. Random search is applied to the architecture search as our baseline and additionally we explore a simple and effective synchronous optimization to reduce the cost of time and computational budgets for search process.

*Southeast University. E-mail:yangsenius@seu.edu.cn

Submission Draft for 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

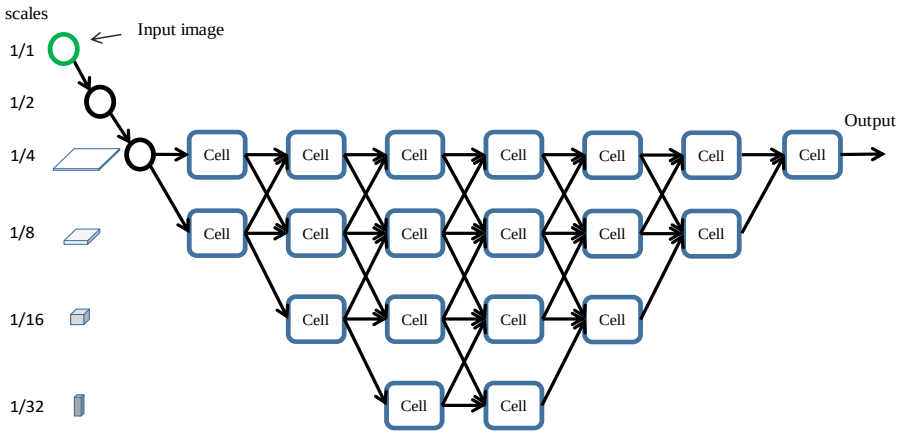

Figure 1: Compact fabric architecture.

The architecture mentioned above is designed to handle pose estimation task. We divide a whole body into multiple parts representations by exploiting the compositonality of body and use proposed cell-based neural fabric architecture to predict each keypoint in part representation. Heatmap representation is usually as response maps of keypoints at each pixel position, we argue that such single scalar value in pixel position is still inadequate to encode local feature of parts. Ambiguity probably occurs between image feature maps and groudtruth heatmaps if some severely occluded invisible keypoints labeled via inference, which making pure supervised learning hard to fit the groundtruth. Inspired by capsules whose activity vector can represent implicit properties of a specified entity, we replace scalar by vector in each pixel position to encode meta-feature of parts and keypoints. We analogously describe it as "vector in pixel", so the localization of keypoint is determined by the position of vector whose length (ℓ_2 norm) is the largest.

In summary, our main contributions are as follow: 1)we propose a new solution for pose estimation that exploiting compositionality of body limbs, we predict each keypoint's location by part representation, 2)we design a compact cell-based neural fabric architecture to learn each part representation by neural architecture search strategies, 3)we validate the effectiveness of part representation with pixel in vector for body keypoints localization. With lightweight model and low computing complexity, our method achieve competitive performance on MPII and MS-COCO benchmarks, compared with state-of-the-art methods.

2 Related Work

The design of neural architecture is motivated by multi-scale neural networks and neural architecture search methods. Most of mehtods for NAS take image classification as basic task so that macro-architecture is not their emphasis. Recently, Chen et al use random search to address dense image prediction with Dense Prediction Cell(DPC). Liu et al [DARTS] propose a trellis-like network with two-level hierarchical structure, the proposed cell-based neural fabric is similar but more compact and simple than it. Nekrasov et al [FAST] adopt reinforce learning and auxiliary cells to search neural architecture for semantic segmentation and transfer models to pose estimation task. More and more complex tasks are involved with NAS methods.

2D human pose estimation involved parts parsing and skeleton keypoints localization can be converted into pixel prediction problem. By top-down or bottom-up way, recent methods[1,2,3,4,5] use deep CNN model to implicitly learn global relationship among different body parts. However, due to parts occlusions, variations in viewpoint, body shape, clothing, lighting and interference from scene, such methods are incapable of capturing complex local feature of body parts. Compositional models [1,2,3,4] represent human body as a hierarchy of parts and subparts to tackle aforementioned problems. Based on top-down method, our part representation exploit the advantage of compositionality of a single body pose.

Our part representation wiht vector in pixel is also motivated by embedding or vector representation introduced in [PAF, AE] for pose estimation. Newell et al [AE] proposes associative embedding to group body keypoints, Papandreou et al [PersonLab] use geometric embedding representation to predict offset vectors of keypoints. Cao et al [PAF] use part affinity vector field to supervise the part prediction. In addition, Hinton et al [Matrix Capsule] use matrix with extra scalar to represent an entity. Sabour et al [Dynamic Capsule] propose activity vector, its length can present existing probability and its orientation represent the instantiation parameters. Our vector in pixel method is inspired by activity vector in [Dynamic Capsule].

3 Problem Formulation

Based on top-down method, 2D human pose estimation aims to locate K keypoints coordiantes $S = \{(x_i, y_i)|x, y \in R, i = 1, 2, ..., K\}$ of body joints(e.g., shoulder, wrist, knee, etc). Using prior knowledge of the kinematics of body limbs, we convert the entire implicit spatial relationship of body into P part representations such as head-shoulder part, left upper arm part, left lower arm part, ... and then we accordingly desgin P subnetworks but sharing backbone to separately predict keypoints location subset s ($s \subseteq S$) whose element belongs to corresponding part. Vector in pixel method is introduced to encode more local feature of body part and the prediction of specified keypoint loaction is determined by the location of vector $\vec{v}$ whose length is the largest.

Next, we demonstrate how to design a compact cell-based neural fabric as choice of subnetworks and backbone, and we explore random search and gradient-based architecture search method to achieve a lightweight model (see section 4.1). In section 4.2, we show how to apply vector in pixel method for part representation to predict keypoints locations and explain how it alleviate the feature ambiguities.

4 Approaches

4.1 Architecture Detail

4.1.1 Compact cell-based neural fabric

$Cell$ is a repeatable unit across different layers and scales of the whole architecture. Illustrated in figure 2, it receives mixed outputs from 3 previous cells as its single input node I and it has H hidden nodes as it hidden states. Each hidden node h_j is connected by a directed edge with each element of candidate nodes set $cn = \{h_0, h_1, h_2, ..., h_{j-1}\}$ ($h_0 = I, j = 1, 2, ..., H$). Continuous Relaxation([DARTS]) method is adopted to represent each directed edge. For each h_j is computed by:

$$h_j = \sum_{i=0}^{j-1} \sum_{o \in \mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} h_i \tag{1}$$

where $\mathcal{O}$ is the set of candidate operations and $\alpha_o^{(i,j)}$ means the associated weight for each operation $o \in \mathcal{O}$ in edge $h_i \rightarrow h_j$. The continuous search space is denoted as variable $\alpha = \left\{\alpha_o^{(i,j)}\right\}$. We concatenate all hidden nodes together and use conv 1×1 to get an independent output node O.

Follow common practice for pose estimation, we use a two-layer convolutional "stem" structure to reduce the resolution to 1/4 scale, consecutively begin with first layer of cells and then "weave" the whole neural fabric in a homogeneous way. In order to get a higher resolution to locate body keypoints coordiantes, we use the cell's output in final layer and 1/4 scale as the part representation(See Section 3.2).

For a specified $Cell_{l,s}$ in scale s and layer l in neural fabric architecture, it receives 3 outputs from previous $Cell_{2s,l-1}$, previous $Cell_{s,l-1}$ and previous $Cell_{\frac{s}{2},l-1}$ [2] with associated weight

[2]For cells in first layer, $Cell_{\frac{1}{4},1}$ and $Cell_{\frac{1}{8},1}$, they recieve only from the stem's output. And for $Cell_{s,l}$ in 1/4 scale or in smallest scale in its current layer, such as $Cell_{\frac{1}{16},2}, Cell_{\frac{1}{4},6}$, it may have no previous cell above or previous cell below or previous parallel cell. In this case, we will copy one of candidate inputs.

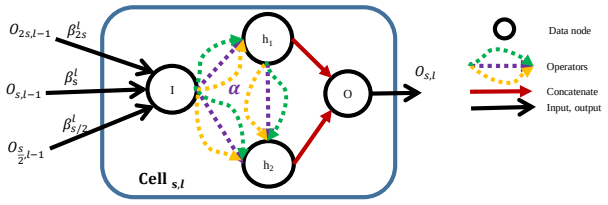

Figure 2: Structure inside the Cell

$\beta^l_{2s}, \beta^l_s, \beta^l_{\frac{s}{2}}$.These three associated weight are normalized exponentially like α to control different connection strength between different cells.

To sum up, we define a general form of homogeneous local connectivity pattern inside the cell:

$$O = \text{Cell}\left(O_{2s}, O_s, O_{\frac{s}{2}}; w, \alpha, \beta, \theta\right) \tag{2}$$

Equation 2 above has omitted the subscript, and in Equation 3 we concretely define the forward calculation inside the cell at the l layer and s scale of compact neural fabric architecture :

$$O_{s,l} = \text{Cell}_{s,l}\left(O_{2s,l-1}, O_{s,l-1}, O_{\frac{s}{2},l-1}; w_{s,l}, \alpha_{s,l}, \beta_{s,l}, \theta_{s,l}\right) \tag{3}$$

where $w_{s,l}$ represents the weights of operations, $\alpha_{s,l}$ and $\beta_{s,l}$ encode architecture search space inside the Cell. $\theta_{s,l} = (H, C, \mathcal{O})$ is the hyperparameter of cell, where H is the number of hidden nodes inside the Cell, C is the channel factor for each node to control the model capacity and $\mathcal{O}$ is the set of candidate operations.

4.1.2 Optimization

Given a hyperparameter Θ for the whole neural fabric, the weights $w = \{w_{s,l}\}$ and architecture $\alpha = \{\alpha_{s,l}\}, \beta = \{\beta_{s,l}\}$ need to be optimized. We follow the principle of one-shot architecture search [DARTS], to search for a repeatable structure whose $\alpha_{s,l}$ share weights across the architecture. Therefore, we assume $\alpha = \alpha_{s,l}$ for all cells . In our design, β controls the connection strength between layers and scales in macro level.

DARTS converts architecture search problem into a bilevel optimization problem and solves it by gradient descent with $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$. Original train set $train_o$ is split into $train$ and val and the $\mathcal{L}_{val}$ serves as performance validation to produce architecture gradient. However, training for weights of operations is partial training and the final architecture must be trained again from scratch. In addition , gradient-based method with second-order (or first-order approximation)consumes much time and GPU memory for pose estimation due to high resolution representation and dense connections in architecture. Although such gradient-based method is effective, it is still time-consuming and restricted by computing resources for dense prediction.

Random Search Random search can be seen as a powerful baseline [Random wiring] for neural architecture search or hypermeter optimization. It is conducted in [Random wiring] as well with competitive results compared with gradient-based method. Therefore, we make it as our baseline neural architecture search to validate the performance of cell-based neural fabric. We randomly initialize values of α, β by standard normal distribution and fix them in the whole training process. 5 experiments with different pseudo random seeds are implemented and the mean and standard deviation are reported in section.

Synchronous Optimization We explore a more simple way based on gradient descent. The α and β are registered to model's parameters, synchronously optimized by the same optimizer of the weights w, so α, β are upadted by $\nabla_{w,\alpha,\beta}\mathcal{L}_{train_o}$ in a single step of gradient descent. With no extra validation performance in [DARTS], the final values of continuous coefficient α, β has seen all training samples and then it do not need be recovered into discrete architecture by $argmax$ operation. Results show that this synchronous gradient-based optimization is effective as well as random search method (see).

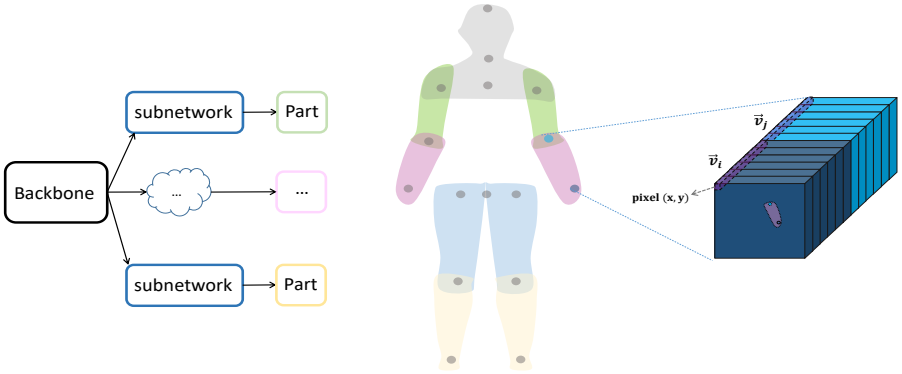

Figure 3: Body part representation with pixel in vector

4.1.3 Extending into Subnetwork or Backbone

This cell-based fabric architecture is very flexible and easy to extend into different layers and scales for high and low resource use cases, determined by a group of hyperparameters $\Theta = (L, scales, H, C, \mathcal{O})$. Given a total layers L and scales types $= \{1/4, 1/8, ..., 1/2^s\}\,(s \geq 2)$, a cell-based neural fabric architecture can be constructed. It is mentioned that the smallest scale is 1/32, and then L should ≥ 6 to ensure that 3-th layer has cell in 1/32 scale. In addtion, fabric architecture backbone is attained by reserving stem and first $m(\geq 3)$ layers and discarding latter $L - m$ layers to produce feature pyramid with 4 types of output in all scales. Likewise, subnetwork is attained by reserving latter $n(\geq 3)$ layers and discarding first $L - n$ layers to receive feature pyramid from backbone.

Then, we use a backbone and P subnetworks to predict P part representation of human pose.

4.2 Part Representation with Vector in Pixel

Following the top-down method of pose estimation, we estimate human pose with single person proposal from detector. Given a person proposal, let $T_p \in R^{N \times H \times W \times d}$ be its p-th body part representation, where N is the number of keypoints belonging to this part, H and W are the height and width of part representation and d is the dimension of vector in each pixel position. For i-th keypoints of T_p, vector in pixel (x, y) is denoted as $\vec{v}_{i,x,y} = T_p\,(i, x, y)$, simplified as $\vec{v}$. Note that the dimension d of vector is set to 8 by default.

We relax scalar value into a high dimension vector to encode meta-feature in each pixel location, expecting it to implicitly capture more local feature information of keypoint. Inherent to the characteristics of encoding location, $\vec{v}$ also can represent existing probability of keypoints by using Squashing Function [Dynamic Capsule] to normlize its $L2$ norm to $[0, 1]$. Concretely, for i-th keypoint of p part, the squashed vector $\vec{v}_s$ in position (x, y) is computed by:

$$\vec{v}_s = \frac{\|\vec{v}\|^2}{1 + \|\vec{v}\|^2} \frac{\vec{v}}{\|\vec{v}\|} \tag{4}$$

where $\|\vec{v}_s\|$ exactly represents i-th keypoint's existing score in position (x, y). The longest $\vec{v}_s$'s position $(\bar{x}, \bar{y})$ will be regarded as keypoint localization in inference and predicted score maps are achieved by all part presentations. Loss is computed by Mean Square Error (MSE) between the predicted score maps and groundtruth score maps with gaussian peak generated from labeled keypoints position. In some cases, joint like elbow maybe fall into multiple parts, so its final position $(\bar{x}, \bar{y})$ will be summed from predictions of these parts.

The extra advantage of vector presentation is that ambiguity between image feature and groundtruth position can be alleviated due to its weak supervision. In supervised learning, the difficulty of fitting label is usually not under consideration, hard or easy samples of the same category receive same level of supervision. This issue occurs in keypoints localization due to some invisible or occluded keypoint position annotated by inference. See first row and first line of figure 5, the man's left ankle is completely occluded by a dog, but its position is annotated by human. In such case, our method can

handle it because $\|\vec{v}_s\|$ replaces $\|\vec{v}\|$ under supervision and the expected length for $\vec{v}$ in groundtruth keypoint pixel is not directly supervised by absolute numerical value.

5 Experiments

5.1 Implementation Details

We choose 6 types of basic operations as candidate operations $\mathcal{O}$, consisting of: zero(no connection), skip connection, 3×3 depthwise separable convolution, 3×3 atrous convolution, 3×3 average pooling, 3×3 max pooling. The last two pooling operations are discarded for subnetwork. Following common design in pose estimation, 1/32 scale is set as the smallest scale, for cell in $s \in \{1/4, 1/8, 1/16, 1/32\}$ scale of neural fabric architecture, the number of channels of its data node(see figure) is $C \times \frac{1}{s}$ where C is channel factor controlling the parameters size of model. Output from previous $Cell_{2s,l-1}$ is transformed by Conv-BN-Relu mode with 2 stride, output from previous $Cell_{\frac{s}{2},l-1}$ is up-sampling with bilinear interpolation and output directly from previous $Cell_{s,l-1}$ are added together as the single input node of the $Cell_{s,l}$.

5.2 Ablation Study

Dataset MPII Human Pose Dataset is a benchmark for evaluation of pose estimation. The dataset consist of around 25K images containing over 40K people with annotated body joints. Due to test annotations unavailable , all models in ablation study experiments are trained on a subset MPII training set and evaluate on a held validation set of 2958 images following [simple, hrnet]. For single pose estimation of MPII, we have two choices for input size: 256x256 and 384x384. Note that all of experiments in our work adopt the same data augmentation strategies with random rotation range in $[-45°, 45°]$, random scale range in $[0.7, 1.3]$ and random flipping with 0.5 probability. Flip test in inference is used.

Body Part Representation Modes We design three modes of part representation. The number of parts P is 1, 3, 8 for different modes. If $P = 1$, the single part representation is treated as global implicit relationship for body pose. If $P = 3$, body pose is divided into head part, upper limb part and lower limb part. If $P = 8$, there are parts including head-shoulder, left upper arm, left lower arm, right upper arm, right lower arm, upper legs, left lower leg and right lower leg (shown in figure 3). All keypoints are associated with corresponding part. We study these three modes and results show in table 2. Mode with $P = 3$ is a trade-off between performance and model parameters due to more parts demanding more subnetworks. We implement our experiments by PyTorch on a single Nvidia Titan Xp GPU. Input size is 256x256, training epoch is 200, we use Adam for weights optimizer with 0.001 initial learning rate, decay at epoch 90,120,150 with 0.25 factor. we use synchronous optimization .

Optimization Strategy For random search strategy, we conduct 3 experiments with different pseudo random seeds, results show that random search is also well-performing. We implement first-order gradient-based optimization method according to [DARTS], which we hold out half of MPII training data as validation for performance estimation of architecture. Another Adam optimizer is adopted to update α, β with 0.003 learning rate and 0.001 weight decay. For synchronous optimization, w, α, β are optimized by Adam. We can see synchronous optimization is effective as first-order gradient-based search method. It is mentioned that synchronous optimization without extra architecture performance estimation needs more discussion about

Vector in pixel To transform final cell's output into part presentation, one way is to use 3d Convolution layer transform it into a 5D tensor $R^{B\times N\times H\times W\times d}$ and another way is to use 2d Convolution layer to $R^{B\times N*d\times H\times W}$ and then reshape it to $R^{B\times N\times H\times W\times d}$. By computing ℓ_2 norm of each vector in it, the final score map $R^{B\times N\times H\times W}$ of part representation can be achieved.

In order to validate the effectiveness and generalization of vector representation method, we apply it in SimpleBaseline based on its official code as well. We explore different choices for dimension of vector with $d = 2, 4, 8, 16$and 3d or 2d convolution choices in final layer of subnetworks. We find that 8-dim with 2d convolution have best performance with a little improvement than official result.

5.3 MPII Single Person Pose Estimation

Table 1: Performance comparisons on MPII validation set

Method	Backbone	Pretrain	Params	Madds	Search Method	Hea	Sho	Elb	Wri	Hip	Kne	Ank	Mean	Mean@0.1
SimpleBaseline	ResNet-50	✓	34.0M	12.0G	×	96.4	95.3	89.0	83.2	88.4	83.9	79.6	88.5	33.9
SimpleBaseline	ResNet-50	✓	34.0M	27.0G	×	96.7	95.8	89.8	84.6	88.5	84.7	79.3	89.1	38.0
HRNet-W32	HRNet	✓	28.5M	9.5G	×	97.1	95.9	90.3	86.4	89.1	87.1	83.3	90.3	37.7
AuxiliaryCell	MobileNet-v2	✓	2.6M	-	RL	95.9	94.4	86.3	80.2	87.1	81.2	75.9	86.5	31.4
Our	Fabric	×	10.5M	3.3G	gradient(sync)	96.0	94.3	86.9	81.4	85.8	80.1	75.3	86.4	35.1
Our	Fabric	×	21.2M	6.7G	gradient(sync)	96.6	94.8	88.8	84.2	87.6	82.7	77.9	88.1	37.2
Our	ResNet-50	✓	27.9M	6.2G	gradient(sync)	96.9	95.5	88.9	83.8	88.8	84.0	79.4	88.8	36.3
Our	ResNet-50	✓	27.9M	9.7G	gradient(sync)	96.8	95.1	89.6	85.0	88.7	85.4	80.5	89.2	39.1
Our	MobileNet-v2	✓	3.3M	1.1G	gradient(sync)	96.7	94.2	87.2	81.8	87.6	81.4	76.3	87.1	34.3
Our	MobileNet-v2	✓	3.3M	1.1G	random	96.7	94.5	87.9	82.8	87.2	81.3	76.2	87.3	35.7
Our	MobileNet-v2	✓	3.3M	1.1G	gradient(first)[3]	96.5	94.6	87.1	82.0	87.2	81.0	76.4	87.1	34.7

Table 2: Performance comparisons on MS-COCO val2017 set

Method	Backbone	Pretrain	Params	Madds	Search Method	AP	AP(0.5)	AP(0.75)	AP(M)	AP(L)	AR
8-stage Hourglass	8-stage Hourglass	-	25.1M	14.3G	-	0.669	-	-	-	-	
CPN+OHKM	ResNet-50	✓	27.0M	6.20G	-	0.694	-	-	-	-	
SimpleBaseline	ResNet-50	✓	34.0M	8.90G	×	0.704	0.886	0.783	0.671	0.772	0.763
SimpleBaseline	ResNet-101	✓	53.0M	12.4 G	×	0.714	0.893	0.793	0.681	0.781	0.771
SimpleBaseline	ResNet-152	✓	68.6M	15.7 G	×	0.720	0.893	0.798	0.687	0.789	0.778
HRNet-W32	HRNet	✓	28.5M	7.10G	×	0.744	0.905	0.819	0.708	0.81	0.798
AuxiliaryCell	MobileNet-v2	✓	2.9M	-	RL	0.659	0.890	0.729	0.631	0.700	0.693
Our	ResNet-50	✓	28.0M	11.4G	sync	0.716	0.885	0.782	0.673	0.800	0.775
Our	Mobilenet-v2	✓	4.1M	2.5G	sync	-	-	-	-	-	-

As for subnetworks, we set all with C=10 and total final layers $L = 3$ (discarding first 3 layers of neural fabric architecture with $L = 6$). If we choose fabric architecture as backbone, C and L have several choices, reported later. In order to make fair comparison with methods using model[1,2,3] pretrained on ImageNet, we also take pretrained ResNet-50 and Mobilenet-V2 as backbone and search for lightweight subnetworks for low resource use cases, so we set parts number $P = 3$ and hidden nodes number $H = 1$ and then the parameters size of the subnetwork for a part is 0.63M. As show in Table, our method achieves competive performance compared with recent state-of-art methods on MPII single person pose estimation valid dataset with less parameters and Multiply-Adds (Madds).(compared with simplebaseline?)(Input resolution 256x256 and 384x384 fo)

Random Search Random search is deemed as a competitive baseline for hyperparameter optimization [baseline, random wire]. We implement a simple random search experiment as well. We randomly initialize values of α, β with Gaussian distribution and fix them in the whole training process ....and then update the w with early-stopping. With multi

Differentiable Architecture Search In gradient-based method, NAS is bilevel optimization problem. By DARTS's method[12], original training dataset $train_o$ is split into $train$ and val, and optimization algorithm includes updating w by $\nabla_w \mathcal{L}_{train}$ and updating α, β by $\nabla_{\alpha,\beta} \mathcal{L}_{val}$. Second-order and first-order approximation can solve this problem. In practice, second-order derivative term will unroll the w consuming much more GPU memory and increase computing complexity with little improvement (see) for this task. So we only use first-order gradient-based search strategy In order to achieve a more lightweight discrete architecture, Darts uses $o^{(i,j)} = \mathrm{argmax}_{o \in \mathcal{O}} \alpha_o^{(i,j)}$ to replace mixed operations for each $h_i \to h_j$edge. In our design, we reserve the continuous architecture α, β instead of inducing them into a discrete architecture and training it again from scratch.

Synchronous Optimization We also explore a more simple and effective way to optimize the w, α, β synchronously. Specially, the α and β are registered to model's parameters identical to the w. Thus, the w, α, β will be updated by $\nabla_{w,\alpha,\beta} \mathcal{L}_{train_o}$ in a single step of gradient descent so as to minimize the loss function.

We make a contrast experiment of first-order gradient-based search method and the synchronous Optimization, results show our synchronous optimization is effective as the former with a little improvement.

Our cell-based neural fabric architecture A is built in macro-level and micro-level. The macroscopic construction method is like convolutional neural fabric and auto-deeplab. The microscopic connection inside the cell is following the common principles in neural architecture search like Darts or DPC(Dense Prediction Cell).(Our architecture is more simple and compact. Then, a group of hyper-parameters specifies the architecture configuration as a choice for subnetwork.)