Supplementary Material for Accurate-PGNet: Learning to Assemble Perceptual Body Parts for Accurate Human Skeleton Establishment

Renjie Zhang, Di Lin, *Member, IEEE*, Xin Wang, George Baciu, *Member, IEEE*, C. L. Philip Chen, *Fellow, IEEE*, and Ping Li, *Member, IEEE*

***Abstract*—This supplementary document provides more details of our method, including algorithm details, searched network architectures and experimental results together with the ground truth images, which could not be fit in the main paper.**

I. Algorithmic Details of Joint Grouping

We use NAS to determine the architecture of Accurate-PGNet. To present the joint grouping clearer, we provide the algorithmic details of architecture search and parameter optimization of Accurate-PGNet in Algorithm 1. Here, we define $grouping\_epoch\_start$ and max_epoch as the epoch when we start the *part grouping* and the maximum epoch for training. Besides the architecture parameters in searchable connections $\mathbb{C}$, we define the set of network parameters as $\mathbb{W}$. And we denote $grouping\_epoch$ as the pruning epoch number. We let $grouping\_epoch\_start < grouping\_epoch < max\_epoch, t \in \{0, ..., T-1\}$.

Algorithm 1: Network Optimization of Accurate-PGNet

```
epoch = 0;
while epoch ≤ grouping_epoch_start do
    Update 𝕎 and ℂ to minimize ℒ_h;
    epoch++;
end
while epoch ≤ grouping_epoch do
    for t = 0 to T-1 do
        foreach 𝔾_m^{t+1} do
            𝒮: Select the main groups 𝔾_⋆^t for 𝔾_m^{t+1};
            ℳ: Select the normal groups 𝔾_†^t for 𝔾_m^{t+1};
        end
        foreach j ∉ 𝔾_i^{t+1}, ∀𝔾_i^t ∈ {𝔾_1^t, ..., 𝔾_N^t} do
            ℐ: Select the inactive groups 𝔾_‡^t, j in 𝔾_‡^t;
        end
        G: Merge H_{⋆,o}^{t,t+1}, H_{†,o}^{t,t+1} and H_{‡,o}^{t,t+1};
    end
    Update the weights in ℂ, 𝕎 to minimize ℒ;
    epoch++;
end
Prune the connections with the zero hidden feature
  map in-between;
while epoch ≤ max_epoch do
    Update the weights in 𝕎 to minimize ℒ_h;
    epoch++;
end
```

II. Implementation Details

In our work, all backbone networks used in our experiments are pretrained on ImageNet [1]. We set the total training epoch as 270, and the architecture searching is processed in the first 75 epochs. And as we described in the paper, we search the architecture of PGB in each stage progressively. After 30 epochs from the search beginning, we start to do pruning, and every 15 epochs, the architecture of PGB in a certain stage will be pruned. It means the $prune\_epoch$ is set as {30, 45, 60, 75}. During searching and training, the batch size is set to 12. For network weight parameters $\mathbf{w}$, we use the Adam optimizer [2] to update them with 1e-4 as learning rate. For optimizing architecture parameters $\mathbb{C}$, we also use Adam optimizer with a fixed learning rate of 3e-3. After the searching, we use a new Adam optimizer to update $\mathbf{w}$ with 1e-4 as initial learning rate, and it is decay at 90^{th}, 120^{th} and 150^{th} epochs with 0.25 factor.

We conduct architecture search seperately on the MPII [3] and MS-COCO [4] datasets. During the network architecture search, we follow [5] to randomly select half of the images in *train* datasets belonging to COCO or MPII sets to update the architecture parameters and the other half is used to update the weight parameters, respectively. The searched architectures are illustrated in Fig. 1

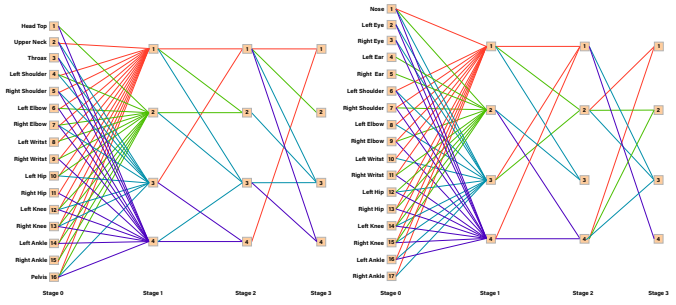
Manuscript received 24 October 2023; revised 8 July 2024; accepted 4 August 2024. This work was supported in part by The Hong Kong Polytechnic University (PolyU) under Grants P0048387, P0042740, P0044520, P0043906, P0049586, and P0050657, and in part by the PolyU Research Institute for Sports Science and Technology under Grant P0044571. *(Renjie Zhang and Di Lin contributed equally to this work.) (Corresponding author: Ping Li.)*

Renjie Zhang, Xin Wang, and George Baciu are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: renjie.zhang@connect.polyu.hk; xin1025.wang@connect.polyu.hk; csgeorge@polyu.edu.hk).

Di Lin is with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China (e-mail: ande.lin1988@gmail.com).

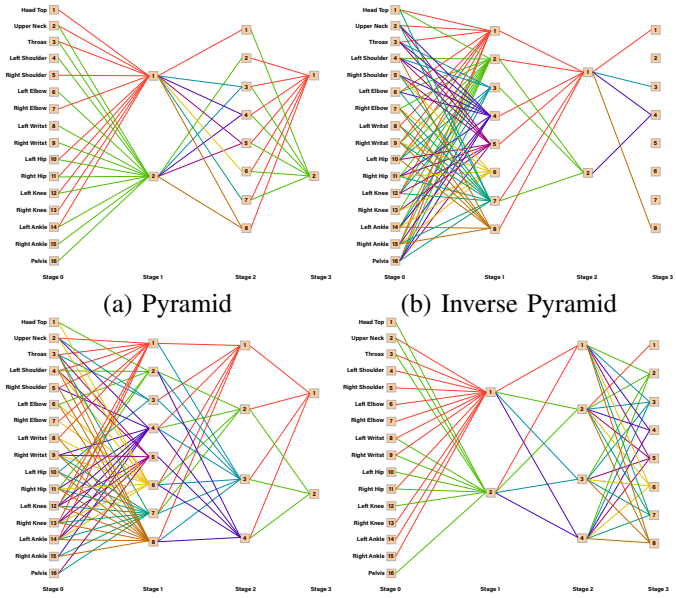
C. L. Philip Chen is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Pazhou Lab, Guangzhou 510335, China (e-mail: philip.chen@ieee.org).

Ping Li is with the Department of Computing, the School of Design, and the Research Institute for Sports Science and Technology, The Hong Kong Polytechnic University, Hong Kong (e-mail: p.li@polyu.edu.hk).



(a) (b)

Fig. 1. The connections between groups at adjacent stages. We train Accurate-PGNet on (a) MPII, and (b) COCO datasets.



(a) Pyramid (b) Inverse Pyramid

(c) Bottleneck (d) Spindle

Fig. 2. The connections between groups at adjacent stages. Here, the (a) pyramid, (b) inverse pyramid, (c) bottleneck, and (d) spindle architectures are used to initialize the architecture search.

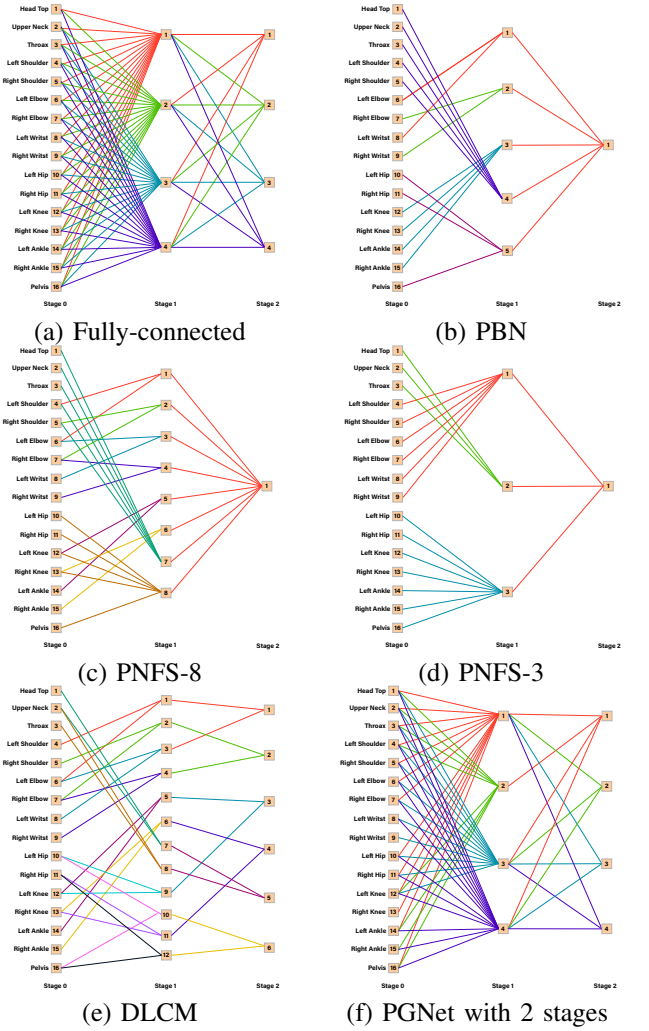
III. Supplementary Description of Experiments

a) Sensitivity to Initial Architectures: In Table III of the main paper, we compare the HPE performances of different initial network architectures. For a fair comparison, we set the number of stages to 3 for different initial architectures. At each stage, the minimum and maximum group numbers are set to 2 and 8. In the column "Groups", we list the number of groups at each stage. In Fig. 1(a), we show the final architecture, which is searched based on the initial plain architecture. In Fig. 2(a)–(d), we use the pyramid, inverse pyramid, bottleneck, and spindle architectures to initialize the architecture search.

b) Comparison with Prescribed Grouping Strategies: In Table V of the main paper, we compare Accurate-PGNet with the prescribed strategies for part groups. In this supplementary material, we provide the network architectures of the compared methods in Fig. 3.

IV. Visual Results

In Fig. 4 of this supplementary material, we present more visual results of different methods on MPII and COCO datasets.



(a) Fully-connected (b) PBN

(c) PNFS-8 (d) PNFS-3

(e) DLCM (f) PGNet with 2 stages

Fig. 3. The architectures of (a) fully-connected, (b) PBN [6], (c) PNFS-8 [7], (d) PNFS-3 [7], (e) DLCM [8], and (f) Accurate-PGNet with 2 stages.

We also provide more visual results of our method in Fig. 5 to validate the effectiveness of our approach.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[2] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[3] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D human pose estimation: New benchmark and state of the art analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3686–3693.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[5] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–13.

[6] W. Tang and Y. Wu, "Does learning specific features for related parts help human pose estimation?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1107–1116.

[7] S. Yang, W. Yang, and Z. Cui, "Searching part-specific neural fabrics for human pose estimation," *Pattern Recognit.*, vol. 128, pp. 108 652:1–108 652:14, 2022.

[8] W. Tang, P. Yu, and Y. Wu, "Deeply learned compositional models for human pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 197–214.

(a) Ground Truth (b) HRNet (c) DARK (d) TransPose (e) Accurate-PGNet

Fig. 4. The part recognition results of HRNet-based methods on MPII and COCO datasets, respectively. Larger dots are the joints with problematic locations. For a clear visualization, we only provide the predicted joints on a person in each image.

Fig. 5. The part recognition results of our method on MPII and COCO datasets.