DynPose: Largely Improving the Efficiency of Human Pose Estimation by a Simple Dynamic Framework

Yalong Xu[1], Lin Zhao[1*], Chen Gong[1], Guangyu Li[1], Di Wang[2], Nannan Wang[2]
[1]PCA Lab, School of Computer Science and Engineering, Nanjing University of Science and Technology
[2]State Key Laboratory of Integrated Services Networks, Xidian University
{yalongxu, linzhao, chen.gong, guangyu.li2017}@njust.edu.cn, {wangdi, nnwang}@xidian.edu.cn

Abstract

Top-down approaches for human pose estimation (HPE) have reached a high level of sophistication, exemplified by models such as HRNet and ViTPose. Nonetheless, the low efficiency of top-down methods is a recognized issue that has not been sufficiently explored in current research. Our analysis suggests that the primary cause of inefficiency stems from the substantial diversity found in pose samples. On one hand, simple poses can be accurately estimated without requiring the computational resources of larger models. On the other hand, a more prominent issue arises from the abundance of bounding boxes, which remain excessive even after NMS. In this paper, we present a straightforward yet effective dynamic framework called DynPose, designed to match diverse pose samples with the most appropriate models, thereby ensuring optimal performance and high efficiency. Specifically, the framework contains a lightweight router and two pre-trained HPE models: one small and one large. The router is optimized to classify samples and dynamically determine the appropriate inference paths. Extensive experiments demonstrate the effectiveness of the framework. For example, using ResNet-50 and HRNet-W32 as the pre-trained models, our DynPose achieves an almost 50% increase in speed over HRNet-W32 while maintaining the same-level accuracy. More importantly, the framework can be generalized to other pre-trained models and datasets without re-training or fine-tuning. Code is available at https://github.com/Aritoria/DynPose.

1. Introduction

Top-down approaches for human pose estimation (HPE) have reached a remarkable level of maturity, with significant improvements in accuracy. Modern architectures like HRNet [32] and ViTPose [37] exemplify this progress. HRNet, which uses multi-scale convolution, provides a significant boost over the baseline [36]. Subsequently, methods like ViTPose introduce attention mechanisms [36], reaching state-of-the-art accuracy. Notably, ViTPose-L achieves an impressive 78.2% AP on the COCO dataset [24].

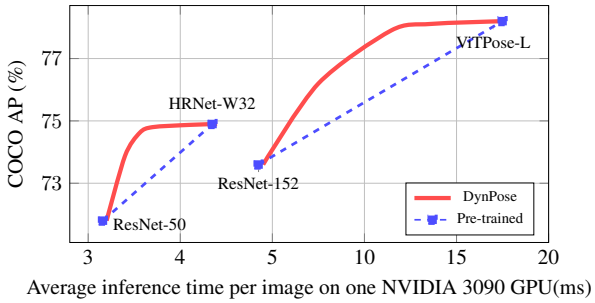

Figure 1. Comparison of DynPose and other classic methods on COCO dataset. The red line on the left shows the trade-off between speed and accuracy of our framework, while the red line on the right illustrates the generalization results.

While current methods excel in accuracy, they often struggle with efficiency [13, 42]. On one hand, top-down approaches are inherently composed of two processes, requiring the cropping of samples from images by object detectors [8] before making pose estimation. The computational cost increases linearly with the number of detection boxes. On the other hand, the increasing complexity of top-down model designs leads to higher inference time.

Recent advances such as lightweight networks [31, 41] and sparse attention methods [1, 5, 16] focus on improving efficiency through innovative designs of the model itself. From another perspective, we discover that the true bottleneck for the slowness of top-down HPE methods may stem from the diversity of pose samples. The diversity arises from two aspects: Pose Diversity and Box Diversity, as illustrated in Fig. 2. Pose Diversity is influenced by whether a pose is common or not, as well as by factors such as occlusion, background clutter and the resolu-

*Corresponding author.

tion of the image where the sample is cropped. Box Diversity, on the other hand, concerns the quality of the bounding boxes [10]. High-quality boxes, with high Intersection over Union (IoU), accurately capture the body and reduce background clutter. In contrast, low-quality boxes often miss limb details and introduce extra background noise. More importantly, there's a significant issue arising from the redundancy of detection boxes. Despite using post-processing techniques like non-maximum suppression (NMS), the detector still generates numerous redundant boxes. For example, there are roughly 140k boxes (detected by Faster R-CNN [10]) for only 6k samples on COCO *val* set. Using one complex model for these diverse samples brings a significant computational waste. For example, when dealing with an easy sample (a common pose with a clean background), ViTPose-L provides the same accuracy as ResNet-50 but takes ten times longer. Therefore, assigning each sample to the most appropriate models will be beneficial.

In this paper, we present a simple yet effective dynamic framework called DynPose, which dynamically allocates resources according to the specific demands of each sample. It features a router and two pre-trained models: one Small Network with lower accuracy but faster inference time, and one Large Network with higher accuracy but slower inference time. The router, which is the core of our design, manages the inference path. It consists of several convolutional and fully connected layers. We supervise the router by designing an optimization strategy that considers the performance gap between the Small and Large Networks, teaching it to evaluate the estimation difficulty of diverse samples. During inference, the router only sends relatively hard samples (those most likely to have accuracy discrepancies between two pre-trained networks) to the Large Network, thereby greatly improving efficiency without sacrificing accuracy. Moreover, with the well-trained router, our framework can generalize to other models and datasets by replacing pre-trained networks. To the best of our knowledge, we are the first to discover sample diversity as a key factor in the efficiency challenge of top-down HPE, and propose a simple dynamic framework to address this.

Our framework achieves a good balance between accuracy and speed, as shown in Fig. 1. The red line on the left shows that when using pre-trained models like ResNet-50 and HRNet-W32, our framework cuts inference time in half while achieving comparable results with HRNet-W32. Meanwhile, the red line on the right shows that directly applying the trained router to ResNet-152 and ViTPose-L still speeds up inference time by around 40%. In summary, our contributions are as follows:

- We discover that sample diversity plays a crucial role in the efficiency issues of Top-Down HPE.
- We introduce a straightforward yet effective dynamic framework that allocates appropriate computational resources to diverse samples to address inefficiencies.
- Experiments show that our framework significantly reduces inference time and generalizes well across various models and datasets.

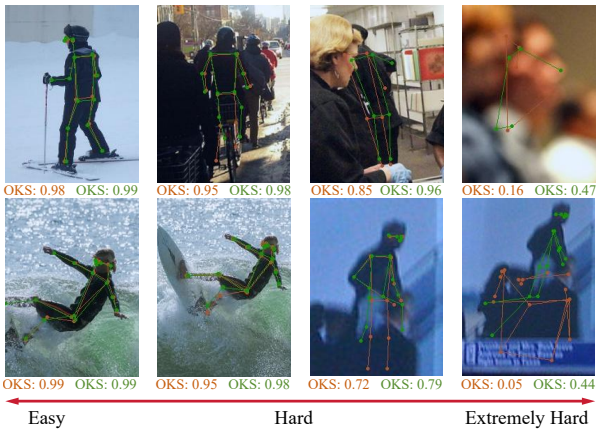

Figure 2. Illustrations of sample diversity. The first row shows variety examples of Pose Diversity. The second row shows the Box Diversity with decreasing IoU. Orange and green annotations represent the predictions of ResNet-50 and ViT-L, respectively.

2. Related Works

2.1. Top-down human pose estimation

In recent years, top-down human pose estimation methods have made significant strides in accuracy. Primarily, backbone is a crucial component of this progress. ResNet [36] establishes the baseline for the field with commendable results and spawns several variants [11, 44]. Following this, HRNet [32] leverages multi-level features and deeper network structures to achieve even greater performance. Models like Swin [26] and ViTPose [37], which utilize attention mechanisms, have the advantage of capturing long-range information, elevating accuracy to state-of-the-art levels. Beyond advancements in backbones, some research focuses on designing more effective loss functions. For example, SimCC [22] converts the heatmap from 2D to 1D and subdivides each pixel into smaller segments to reduce quantization errors. RLE loss [19] improves accuracy by training the network to learn from a realistic keypoint distribution. Additionally, other works boost performance through sophisticated algorithm designs. SCAI [15] employs intricate architectural designs, enabling the model to continually refine its outcomes. Moreover, Poseur [28] introduces DETR [2] decoder to achieve superior results.

While the methods mentioned above improve accuracy, they highlight the lingering issue of slow inference speed in top-down HPE. To address this, some approaches focus on shrinking the backbone. For example, Lite-HRNet [41] retains the core design of HRNet while significantly

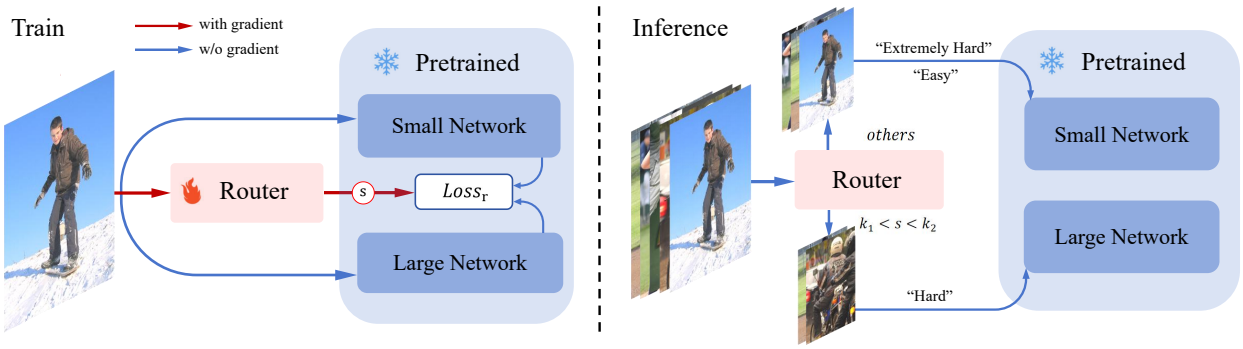

Figure 3. The architecture of DynPose. Our framework includes a Router for scoring samples and two pre-trained models that differ in accuracy and computational cost. During inference, the image is passed through the Router to obtain a score s, which dynamically determines the inference path. k_1 and k_2 are the thresholds used to distinguish "Easy", "Hard" and "Extremely Hard". We route "Hard" samples to Large Network. "Easy" and "Extremely Hard" are send to Small Network.

reducing the number of parameters. Similarly, the MobileNet [31] and ShuffleNet [45] series use fewer computational resources to achieve cost-effective results. Apart from the backbone, some methods employ innovative algorithm designs to improve efficiency. RepVGG [7] introduces a weight-sharing mechanism to reduce FLOPs during inference. SHaRPose [1] designs a sparse attention method that helps the network concentrate on key areas, improving both accuracy and speed. Additionally, in the field of distillation, FPD [43] achieves parameter compression through distillation between a large model and a small model, while SDPose [4] employs a cyclical distillation method that enhances both speed and accuracy. These existing methods mainly aim to speed up inference by optimizing the complexity of the models. From a different perspective, our approach views sample diversity as a crucial factor affecting the speed of inference and designs a dynamic framework to improve efficiency.

2.2. Dynamic neural networks

Dynamic neural networks [14, 38] are often used to improve efficiency [9] and are relatively mature in image classification. They often employ an early exiting strategy, where the result exits immediately if an earlier layer reaches a high confidence level, thereby bypassing subsequent computations. Branchynet [33] is one of the earliest networks to employ this strategy. MSDNet [12] advances this by incorporating multi-level features. Subsequent researches [17, 39] have further refined this approach. Additionally, DVT [34] integrates transformer blocks and improve both efficiency and performance.

Additionally, another class of dynamic methods focus on predicting inference paths based on features. For example, BlockDrop [35] enhances inference efficiency by leveraging redundancy between residual blocks. It predicts a series of parameters from the features to dynamically decide which blocks to skip. More recently, DynamicDet [25] extends dynamic networks to object detection tasks by introducing a routing module to decide whether to execute the output layer or forward the sample to subsequent blocks. Our framework extends dynamic networks to HPE. Due to its unique design, we only need to train a lightweight router to control the inference path for each sample, and the router is decoupled from the prediction networks. Thus, we keep efficient throughout both training and inference.

3. Methodology

3.1. Overall architecture

Our framework dynamically allocates appropriate computational resources to diverse samples, thereby speeding up inference. The overall structure consists of a router and two pre-trained HPE models, as shown in Fig. 3. The router controls the inference path for diverse samples. The Small Network (lower performance, faster speed) and the Large Network (higher performance, slower speed) are replaceable and used to generate results.

For an image I, we first use the router to predict a score $s \in (0, 1)$ as

$$s = Router(I), \tag{1}$$

which reflects the estimation difficulty of the pose sample. It is influenced by both Pose Diversity and Box Diversity. For example, a sample with a common pose, clean background and a high-quality detection box results in a low s, indicating easier estimation. In contrast, a sample with occlusions and other complicating factors, along with low-quality detection boxes, correspond to a higher s.

Based on the predicted score s and the performance of Small and Large Networks, we categorize the samples into three groups: "Easy", "Hard" and "Extremely Hard". "Easy" samples yield accurate results with both Small and Large Networks. "Hard" samples can only be handled by

the Large Network, while the Small Network struggles. For "Extremely Hard" samples, neither of them performs well, as these samples also exceed the Large Network's capabilities. It is worth noting that the majority of "Extremely Hard" samples are caused by redundant or incorrect detection boxes generated by the object detector, making them meaningless and challenging for accurate pose regression. As exemplified in Fig. 2, although the OKS values predicted by the Large Network are higher than those predicted by the Small Network, they often fall below 0.5 (generally indicates a false prediction).

From the aforementioned analysis, our inference strategy is as follows:

$$\mathrm{y} = \begin{cases} Net_l(I) & \text{if} \quad k_1 < s < k_2 \\ Net_s(I) & \text{else} \end{cases}, \tag{2}$$

where Net_s and Net_l refer to the Small Network and Large Network, respectively. k_1 and k_2 are thresholds that classify samples into three categories. Intuitively, we feed both "Easy" and "Extremely Hard" samples into the Small Network since the performance difference is negligible for these cases, while using the Large Network would incur higher inference costs. For the remaining "Hard" samples, we use the Large Network to ensure overall accuracy.

3.2. Lightweight router

The router is the core of our framework. It processes each sample to predict a estimation difficulty score s. We design the router to be lightweight to keep it efficient.

We first perform dimensionality reduction on the input image. Specifically, for an image $I \in \mathbb{R}^{3\times H\times W}$, we apply average pooling across the channel dimension and utilize max pooling to reduce resolution, transforming I into $F_0 \in \mathbb{R}^{1\times \frac{H}{2}\times \frac{W}{2}}$.

Next, we process F_0 using four carefully designed convolution-based layers. They can be described as

$$\mathrm{F}_i = \begin{cases} SGE(C(F_{i-1})) & i \in \{1,3\} \\ PConv(C(F_{i-1})) & i \in \{2,4\}, \end{cases} \tag{3}$$

where $C()$ denotes a 3×3 convolution block that includes activation and normalization. $PConv()$ [3] is an efficient convolution operation. To capture long-range spatial information, we employ a lightweight attention mechanism, SGE [20].

Finally, we flatten F_4 to obtain F, which is processed through two fully connected layers with activation functions to get the result of s:

$$s = \sigma(W_2(\delta(W_1F + b_1)) + b_2), \tag{4}$$

where δ, σ denote the ReLU and Sigmoid activation functions respectively. W and b are the trainable parameters in the fully connected layers.

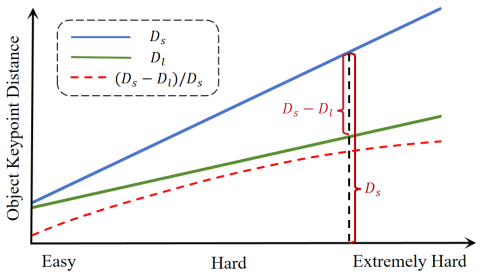

Figure 4. A qualitative analysis of estimation difficulty and OKD. As the estimation difficulty increases, the OKD for both Small and Large Networks (D_s and D_l) grows. But D_s grows much faster than D_l, and we design (D_s-D_l)/D_s to indicate the score of estimation difficulty.

Notably, the computational cost of router is negligible compared to that of Small and Large Networks. For 256×192-sized input images, its FLOPs reaches only 0.19G.

3.3. Optimization strategy

In this section, we provide an optimization strategy for training the router. We aim to supervise the score s enabling it to evaluate the estimation difficulty of each sample. We intend to use the difference in performance between the Small and Large Networks to reflect the estimation difficulty. Directly using Object Keypoint Similarity (OKS) [24] discrepancy is an option, as OKS measures the similarity between predicted keypoints and ground truth.

However, we find that the negative exponential operation in OKS significantly narrows the prediction gap between the networks, especially for challenging samples. To better emphasize the differences between the Small and Large Networks and provide more effective supervision for the router, we convert OKS into Object Keypoint Distance (OKD) by omitting the negative exponential operation as

$$D = \frac{\sum_{i=1}^{17} d^i \rho^i v^i}{\sum_{i=1}^{17} v^i}, \tag{5}$$

where v represents keypoints' visibility, and ρ denotes the hyperparameters in OKS. d is the Euclidean distance between the predicted and ground truth keypoint locations. The variable i denotes the i-th keypoint.

Let D_s and D_l represent the OKD of the Small and Large Networks, respectively. We observe a large number of samples during training and find that they generally follow the relationship shown in Fig. 4. The Large Network has much stronger prediction capabilities. As estimation difficulty increases, D_l rises slowly. However, the Small Network, which is less robust, sees a quick increase in D_s with growing estimation difficulty. The increase in D_s is much faster than that of D_l. Therefore, we use the ratio of $\frac{D_s - D_l}{D_s}$ to represent the difficulty of the samples and we design the

loss function as

$$Loss_r = (s - \frac{D_s - D_l}{D_s})^2. \quad (6)$$

Additionally, D_s will not be zero due to unavoidable errors.

3.4. Inference strategy

In this section, we introduce how to adaptively process diverse pose samples using the most suitable network during inference. To achieve this, the core problem is determining the thresholds k_1 and k_2 of the predicted score s. This strategy allows us to adjust the testing sample split ratio to the Large Network, enabling various trade-offs between speed and accuracy.

First, each testing sample is processed by the router to obtain a score s, and we can sort all the scores to get $S_{all} = \{S^1_{all}, S^2_{all}, \ldots, S^n_{all}\}$, where n denotes the number of testing samples. Here, S_{all} is arranged in an ascending order. We set k_1 to a relatively fixed value as

$$k_1 = S^1_{all} + \theta(S^n_{all} - S^1_{all}), \quad (7)$$

where θ is a hyperparameter. To balance accuracy and speed, k_2 can be set on the desired proportion of data to the Large Network. For instance, if we aim for $p\%$ of the data to be processed by the Large Network, we find the S^j_{all} which is the closest score to k_1 in S_{all} and then set threshold k_2 as

$$k_2 = S^{j+\lfloor n \times p\% \rfloor}_{all}. \quad (8)$$

Generally, k_1 is set to a relatively small value and $(j + \lfloor n \times p\% \rfloor) < n$ can be satisfied.

With k_1 and k_2, the testing samples can be divided into three parts: "Easy", "Hard" and "Extremely Hard". During inference, "Hard" samples are routed to the Large Network and their predicted score s satisfy $k_1 < s < k_2$. Other samples are processed by the Small Network. Note that "Extremely Hard" samples are handled by the Small Network, as even the Large Network cannot produce good results for these samples. Based on this strategy, we achieve variable-speed inference for any pair of Small and Large Networks.

4. Experiments

4.1. Experimental setups

Our experiments are conducted on MMPose [6] and use the COCO 2017 Keypoint Detection [24] benchmark. We train all models on the $train$ set with approximately 100K pose samples and report results on the val set. The Average Precision (AP) metric is used for evaluation. For selecting the Small and Large Networks, we focus on ensuring their performance and speed differences. They can be adjusted flexibly according to the requirements. In this paper, we use the classic ResNet-50 and HRNet-32 as the default combination unless otherwise stated. The training process uses the Adam optimizer with a learning rate of 1×10^{-4} and a batch size of 1. During inference, we utilize the commonly used person detection results from Faster R-CNN [29]. For k_1 and k_2, we set θ to 0.1 and the expected split ratio to Large Network as 50% by default. Speed performance is measured on one NVIDIA 3090 GPU.

Model	AP_{50}	AP	FPS↑	FLOPs↓
ResNet-50	89.8	71.8	313	5.45G
Dyn-R50-H32/25	90.3	73.5	297	6.21G
Dyn-R50-H32/50	90.6	74.8	270	6.76G
Dyn-R50-H32/75	90.6	74.8	252	7.32G
HRNet-W32	90.6	74.9	223	7.69G
ResNet-101	90.4	72.8	261	9.01G
Dyn-R101-SL/25	90.5	74.0	151	17.19G
Dyn-R101-SL/50	90.6	74.3	106	25.18G
Dyn-R101-SL/75	90.6	74.3	81	33.16G
Swin-L	90.6	74.4	66	40.96G
ResNet-152	90.4	73.6	208	12.75G
Dyn-R152-VL/25	90.9	76.9	132	24.42G
Dyn-R152-VL/50	91.4	78.0	93	35.90G
Dyn-R152-VL/75	91.4	78.0	73	47.39G
ViTPose-L	91.4	78.2	59	58.68G

Table 1. The results of DynPose on the COCO benchmark using different combinations of Small and Large Networks for training.

4.2. Efficiency improvement of DynPose

To validate effectiveness of our framework, we train the router using three sets of Small and Large Networks combinations. We use concise notations to represent different combinations and add suffixes like "/50" to indicate data split ratios. For instance, "Dyn-R50-H32/50" signifies that we train the router with ResNet-50 and HRNet-W32. During inference, we direct 50% samples to the Large Network. We also select ResNet-101 with Swin-L and ResNet-152 with ViTPose-L for the other two sets, abbreviate as "Dyn-R101-SL" and "Dyn-R152-VL", respectively. We compute the final FLOPs by multiplying the FLOPs of the Small and Large Networks by their respective split ratios, then adding the router's overhead (0.19 GFLOPs). The final results, shown in Tab. 3, demonstrate that we can significantly reduce FLOPs and FPS while maintaining performance on par with the Large Network for each group. For example, compared to ViTPose-L, Dyn-R152-VL maintain an AP of 78.0% while increasing FPS from 59 to 93 and decreasing FLOPs from 58.68G to 35.90G.

Analysis of efficiency improvement. We statistically analyze the estimation difficulty scores of samples with detection boxes and plot the distribution, as shown in Fig. 5. The significant redundancy of detection boxes (6K real sam-

Method	AP_{50}	AP_{75}	AP_M	AP_L	AR	GFLOPs	AP
TokenPose-S*[21]	88.7	79.0	68.3	78.5	77.0	4.7	71.8
TokenPose-B*[21]	89.5	80.2	70.1	79.8	78.7	5.2	73.2
PPT-S*[27]	87.7	76.8	66.1	76.7	75.1	2.0	69.8
PPT-B*[27]	89.5	80.8	70.3	79.8	78.8	4.7	73.4
OKDHP-2HG[23]	91.5	79.5	69.9	77.1	75.6	25.5	72.8
DistilPose-L[40]	89.9	81.4	71.0	81.8	79.8	10.3	†74.4
SDPose-S*[4]	89.5	80.4	70.1	80.3	78.7	4.7	‡73.5
Dyn-M2-H48(ours)	90.6	82.1	71.2	81.8	80.3	8.8	†75.0(↑0.6%)
Dyn-M2-H32(ours)	90.5	81.5	70.6	81.0	79.7	4.8	‡74.3(↑0.8%)

Table 2. Comparison with the state-of-the-arts small-scale and distillation methods on COCO val set. We construct our model using the classic HRNet and MobileNet architectures. The abbreviation for MobileNet-V2 is M2, while HRNet-W32 and HRNet-W48 are referred to as H32 and H48, respectively. * means the results from SDPose [4]. † and ‡ represents the data pair for comparison.

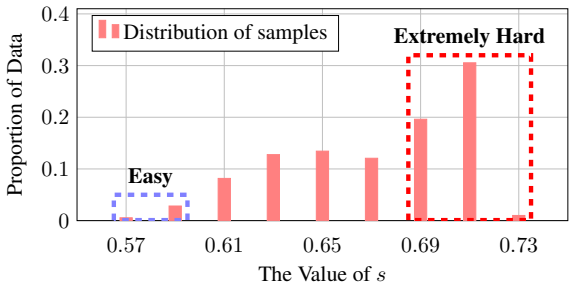

Figure 5. The predicted score distribution from the router of samples with detection boxes. The areas within the dashed boxes indicate the scores for "Easy" and "Extremely Hard" samples.

ples corresponding to 140K boxes) leads to a large number of invalid pose samples. This is because the detection boxes for these samples have relatively low IoUs. When the split ratio is 50%, k_2 is set to 0.69. We find that approximately 50% of the samples accumulate within a narrow range of (0.69, 0.72), corresponding to "Extremely Hard" samples. Our model achieves significant efficiency by routing these samples to the Small Network.

4.3. Comparison with the state-of-the-arts

As shown in Tab. 2, DynPose achieves superior performance compared to other small-scale and distillation methods. Specifically, we compare our method primarily with the small-scale models TokenPose [21] and PPT [27], as well as distillation-based methods OKDHP [23], DistilPose [40] and SDPose [4]. Notably, SDPose combines the advantages of small models and distillation, representing the state-of-the-art. Dyn-M2-H32 achieves 74.3% AP with 4.8 GFLOPs, which outperforms SDPose-S [4] in accuracy by 0.8% while maintaining a comparable FLOPs.

Net_{small}	Net_{large}	FPS	FLOPs	AP
MobileNet V2	-	415	1.58G	64.8
ShuffleNet V2	-	457	1.37G	60.2
ResNet-50	-	313	5.45G	71.8
-	HRNet-W48	161	15.75G	75.6
MobileNet V2	HRNet-W48	222	8.86G	74.9
ShuffleNet V2	HRNet-W48	238	8.75G	75.1
ResNet-50	HRNet-W48	210	10.79G	75.4
-	Swin-L	66	40.96G	74.4
MobileNet V2	Swin-L	115	21.46G	73.9
ShuffleNet V2	Swin-L	123	21.36G	73.8
ResNet-50	Swin-L	108	23.40G	74.3
-	ViTPose-L	59	58.68G	78.2
MobileNet V2	ViTPose-L	110	30.32G	77.5
ShuffleNet V2	ViTPose-L	119	30.22G	77.6
ResNet-50	ViTPose-L	98	32.26G	78.0

Table 3. Generalization results across different models on COCO. The parameters of router directly comes from Dyn-R50-H32 without any fine-tuning. By default, we set the data split ratio to 50%.

4.4. Strong Generalization Ability

Generalization across models. We present the results of directly transferring the router parameters from Dyn-R50-H32 to other models. For Small Network, we select ResNet-50 and two lightweight networks, MobileNet-V2 and ShuffleNet-V2. For Large Network, we use models like HRNet-W48, Swin-L and ViTPose-L. By combining these networks in different configurations, we obtain the results shown in Tab. 3. For instance, with the combination of ResNet-50 and ViTPose-L, we maintain an AP of 78% and increasing FPS from 59 to 98.

Generalization across datasets. We conduct general-

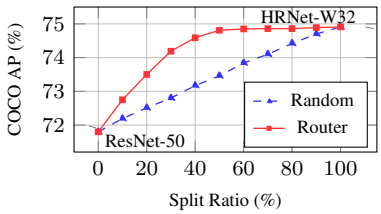

Figure 6. Comparison of router controlled and random-path inference strategy with different split ratios.

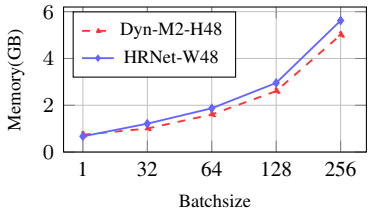

Figure 7. Comparison of memory usage. The dashed line indicates the memory usage of our model, where p is set to 50%.

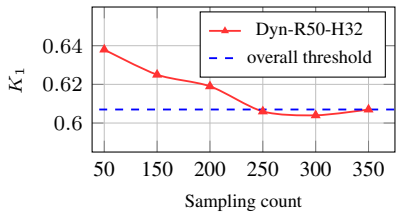

Figure 8. Comparison of the overall threshold K1 (blue) on COCO *val* and the threshold with minimal sampling (red).

Model	AP_{50}	AP	FPS	FLOPs
ResNet-50	80.8	63.7	313	5.45G
Dyn-R50-H32/25	81.8	65.6	297	6.21G
Dyn-R50-H32/50	82.3	66.9	270	6.76G
Dyn-R50-H32/60	82.4	67.3	258	6.98G
HRNet-W32	82.5	67.5	223	7.69G

Table 4. The results of directly transferring the router in Dyn-R50-H32 from COCO to CrowdPose.

Pooling	PConv	SGE	Time(ms)	AP
			0.52	74.64
√			0.10	74.71
√	√		0.09	74.73
√		√	0.13	74.78
√	√	√	0.10	74.81

Table 5. Ablation study of the router designs

ization experiments across different datasets. Specifically, the router Dyn-R50-H32 is directly applied to CrowdPose [18]. As shown in Tab. 4, when the data split ratio is set to 60%, our dynamic framework achieves performance comparable to the Large Network while increasing FPS from 223 to 258. Compared to the 50% split ratio used for COCO, achieving comparable performance on CrowdPose requires directing 60% of the samples to the Large Network and setting θ to 0.3. This is because CrowdPose, designed specifically for crowded scenarios, has a data distribution that diverges significantly from open-world sampling.

4.5. Memory Usage Analysis

Our model includes two networks, which might intuitively imply higher memory usage. However, this is not the case. The GPU memory usage primarily consists of two parts: storage memory (for storing the program and model) and runtime memory (for the memory required during inference). Typically, the Small Network requires less runtime memory than the Large Network. By default, our framework routes 50% samples to the Small Network, resulting in significant runtime memory saving that exceed the storage memory required to store the Small Network. As shown in Fig. 7, Dyn-M2-H48 consumes less memory than HRNet-W48, because MobileNet-V2 requires only about 0.04GB for storage, much less than the saved runtime memory.

4.6. Ablation study

4.6.1. Efficiency of our framework

To validate efficiency, we compare the inference strategy using the router with a random path inference strategy. The results are shown in Fig. 6. The red line represents the results with the router-controlled data split, while the blue dashed line shows the results when samples are randomly routed to the Large Network. Our method outperforms the random-path inference strategy under all split ratios.

4.6.2. Lightweight router design

To validate the lightweight design of the router, we conduct ablation studies on its specific components. First, we discuss the necessity of pooling the image across different dimensions, referred to as Pooling. Next, we explore the PConv and SGE components, detailed in Eq. (3). For the ablation experiments, we remove the Pooling and SGE components, and replace PConv with a standard convolution. The final results are shown in Tab. 5. We observe that Pooling significantly reduces the inference time. PConv and SGE primarily contribute to accuracy improvements.

4.6.3. Robust inference strategy

Our inference process is robust and is primarily determined by two parameters, K_1 and K_2. K_2 depends on p, which determines the proportion of samples that need to be forwarded to the Large Network, and is independent of the model itself. K_1, on the other hand, is a crucial hyperparameter for distinguishing "Easy" and "Hard" samples, and it is mainly determined by θ, which should be a relatively small value. This is because easy samples, such as those with complete and common human poses against clean backgrounds, are rare in open-world datasets. [30] conduct statistics on the COCO dataset, indicating that only

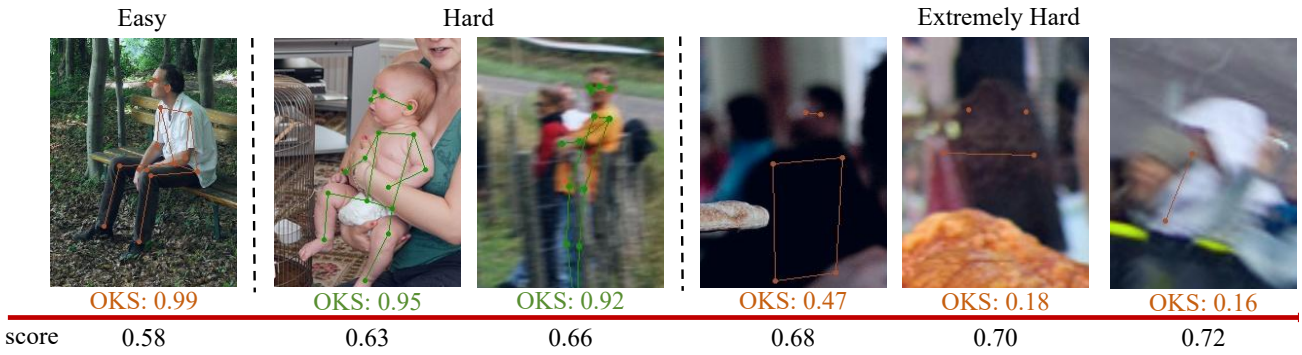

Figure 9. Visualization of diverse pose samples and their corresponding estimation difficulty scores evaluated by the router. The orange and green annotations represent the pose estimation results provided by the Small and Large Networks, respectively.

11.2% of the samples are annotated with complete keypoints (16 or 17). We experiment with θ values ranging from 0.05 to 0.3, as shown in Tab. 6. In comparison, setting θ to 0.1 yields the best performance, which is consistent with the statistical results in [30]. Notably, it can be observed that the choice of θ is not highly sensitive, as good results are achieved within the range of 0.05 to 0.2. Additionally, it requires only minimal sampling to determine an excellent threshold, facilitating application. On COCO *val* set, sampling 200 instances (0.2% of 104K) is sufficient for K_1 determination, as shown in Fig. 8.

In practical scenarios, the parameters can be directly referenced from the COCO *val* set, as COCO is sampled from an open-world environment. Furthermore, as demonstrated earlier, K_1 exhibits good robustness. As shown in Tab. 7, we directly transferred the parameters to *test-dev*. The router sends 49% of the samples to the Large Network and reaches an 73.7% AP. It shows the same data splitting ratio and excellent performance retention. This experiment further validates the robustness of our inference strategy from the perspective of parameter reuse.

4.7. Visualization of results

We visualize pose samples with different scores in Fig. 9, showing an increase estimation difficulty from left to right. The leftmost sample features a clear, unoccluded person, while the second shows partial occlusion and a missing foot. The third and fourth samples highlight challenges from noisy backgrounds, significant occlusion, and lower resolution. The last samples are severely affected, making it difficult even for humans. OKS results show high accuracy for the first sample with the Small Network, while the Large Network handles middle samples well. "Extremely Hard" samples, processed with the Small Network, challenge even the Large Network. In summary, our framework correctly categorizes samples into "Easy", "Hard" and "Extremely Hard", allowing for a rational allocation of computational resources to accelerate inference.

θ	0.05	**0.1**	0.15	0.2	0.3
AP	74.78	**74.81**	74.68	74.48	73.91

Table 6. Setting of the hyperparameter θ on COCO *val*

Model	Set	AP	Split Ratio
Dyn-R50-H32/50	*val*	74.8	50%
ResNet-50	*test-dev*	71.1	-
Dyn-R50-H32/50	*test-dev*	73.7	49%
HRNet-W32	*test-dev*	73.8	-

Table 7. Accuracy and split ratio results of directly transferring thresholds k_1 and k_2 from COCO *val* to *test-dev*.

5. Conclusion

In this paper, we discover that the diversity of pose samples significantly impacts the efficiency of top-down HPE. Hence, an effective yet simple dynamic framework called DynPose is proposed to handle diverse pose samples with the most suitable models, thus achieving a favorable trade-off between speed and accuracy. Specifically, during inference, the router dynamically determines the inference path. Small Network handles both "Easy" and "Extremely Hard" samples, while Large Network processes "Hard" samples. DynPose significantly improves the efficiency of top-down HPE, and additional experiments further confirm its generalization across various models and datasets.

6. Acknowledgments

This work was supported by the National Science Fund of China under Grant 62172222 and the National Key Research and Development Program of China (International Collaboration Special Project, No.SQ2023YFE0102775).

References

[1] Xiaoqi An, Lin Zhao, Chen Gong, Nannan Wang, Di Wang, and Jian Yang. Sharpose: Sparse high-resolution representation for human pose estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 691–699, 2024. 1, 3

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[3] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don't walk: chasing higher flops for faster neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12021–12031, 2023. 4

[4] Sichen Chen, Yingyi Zhang, Siming Huang, Ran Yi, Ke Fan, Ruixin Zhang, Peixian Chen, Jun Wang, Shouhong Ding, and Lizhuang Ma. Sdpose: Tokenized pose estimation via circulation-guide self-distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1082–1090, 2024. 3, 6

[5] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 1

[6] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark, 2020. 5

[7] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021. 3

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[9] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021. 3

[10] 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. 2

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[12] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 3

[13] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. Rtmpose: Real-time multi-person pose estimation based on mmpose. *arXiv preprint arXiv:2303.07399*, 2023. 1

[14] Zequn Jie, Peng Sun, Xin Li, Jiashi Feng, and Wei Liu. Anytime recognition with routing convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):1875–1886, 2019. 3

[15] Zhehan Kan, Shuoshuo Chen, Ce Zhang, Yushun Tang, and Zhihai He. Self-correctable and adaptable inference for generalizable human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5537–5546, 2023. 2

[16] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020. 1

[17] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1891–1900, 2019. 3

[18] 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. 7

[19] Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11025–11034, 2021. 2

[20] Xiang Li, Xiaolin Hu, and Jian Yang. Spatial group-wise enhance: Improving semantic feature learning in convolutional networks. *arXiv preprint arXiv:1905.09646*, 2019. 4

[21] Yanjie Li, Shoukui Zhang, Zhicheng Wang, Sen Yang, Wankou Yang, Shu-Tao Xia, and Erjin Zhou. Tokenpose: Learning keypoint tokens for human pose estimation. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 11313–11322, 2021. 6

[22] Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunxiao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. Simcc: A simple coordinate classification perspective for human pose estimation. In *European Conference on Computer Vision*, pages 89–106. Springer, 2022. 2

[23] Zheng Li, Jingwen Ye, Mingli Song, Ying Huang, and Zhigeng Pan. Online knowledge distillation for efficient pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11740–11750, 2021. 6

[24] 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 *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 1, 4, 5

[25] Zhihao Lin, Yongtao Wang, Jinhe Zhang, and Xiaojie Chu. Dynamicdet: A unified dynamic architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6282–6291, 2023. 3

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer:

Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 2

[27] Haoyu Ma, Zhe Wang, Yifei Chen, Deying Kong, Liangjian Chen, Xingwei Liu, Xiangyi Yan, Hao Tang, and Xiaohui Xie. Ppt: token-pruned pose transformer for monocular and multi-view human pose estimation. In *European Conference on Computer Vision*, pages 424–442. Springer, 2022. 6

[28] Weian Mao, Yongtao Ge, Chunhua Shen, Zhi Tian, Xinlong Wang, Zhibin Wang, and Anton van den Hengel. Poseur: Direct human pose regression with transformers. In *European conference on computer vision*, pages 72–88. Springer, 2022. 2

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 5

[30] Matteo Ruggero Ronchi and Pietro Perona. Benchmarking and error diagnosis in multi-instance pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 369–378, 2017. 7, 8

[31] 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. 1, 3

[32] 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. 1, 2

[33] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016. 3

[34] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in neural information processing systems*, 34:11960–11973, 2021. 3

[35] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8817–8826, 2018. 3

[36] 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. 1, 2

[37] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in Neural Information Processing Systems*, 35:38571–38584, 2022. 1, 2

[38] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2369–2378, 2020. 3

[39] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2369–2378, 2020. 3

[40] Suhang Ye, Yingyi Zhang, Jie Hu, Liujuan Cao, Shengchuan Zhang, Lei Shen, Jun Wang, Shouhong Ding, and Rongrong Ji. Distilpose: Tokenized pose regression with heatmap distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2163–2172, 2023. 6

[41] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A lightweight high-resolution network. In *CVPR*, 2021. 1, 2

[42] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. Hrformer: High-resolution vision transformer for dense predict. *Advances in neural information processing systems*, 34:7281–7293, 2021. 1

[43] Feng Zhang, Xiatian Zhu, and Mao Ye. Fast human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3517–3526, 2019. 3

[44] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2736–2746, 2022. 2

[45] 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. 3