ViPNAS: Efficient Video Pose Estimation via Neural Architecture Search

Lumin Xu[1,2]   Yingda Guan[2]   Sheng Jin[3,2]   Wentao Liu[4]   Chen Qian[4]

Ping Luo[3]   Wanli Ouyang[5]   Xiaogang Wang[1,2]

[1] The Chinese University of Hong Kong   [2] SenseTime Research   [3] The University of Hong Kong

[4] SenseTime Research and Tetras.AI   [5] The University of Sydney

luminxu@link.cuhk.edu.hk   {guanyingda, jinsheng, liuwentao, qianchen}@sensetime.com

pluo@cs.hku.hk   wanli.ouyang@sydney.edu.au   xgwang@ee.cuhk.edu.hk

Abstract

Human pose estimation has achieved significant progress in recent years. However, most of the recent methods focus on improving accuracy using complicated models and ignoring real-time efficiency. To achieve a better trade-off between accuracy and efficiency, we propose a novel neural architecture search (NAS) method, termed ViP-NAS, to search networks in both spatial and temporal levels for fast online video pose estimation. In the spatial level, we carefully design the search space with five different dimensions including network depth, width, kernel size, group number, and attentions. In the temporal level, we search from a series of temporal feature fusions to optimize the total accuracy and speed across multiple video frames. To the best of our knowledge, we are the first to search for the temporal feature fusion and automatic computation allocation in videos. Extensive experiments demonstrate the effectiveness of our approach on the challenging COCO2017 and PoseTrack2018 datasets. Our discovered model family, S-ViPNAS and T-ViPNAS, achieve significantly higher inference speed (CPU real-time) without sacrificing the accuracy compared to the previous state-of-the-art methods.

1. Introduction

Human pose estimation has made impressive progress in recent years with the development of stronger neural networks. Most state-of-the-art models [36, 49, 56] only focus on improving the accuracy, but ignore the computational complexity and real-time performance. However, both accuracy and efficiency are critical for real-world applications of video pose estimation. In this paper, we aim to build a lightweight pose estimator that achieves state-of-the-art performance with significant model complexity reduction.

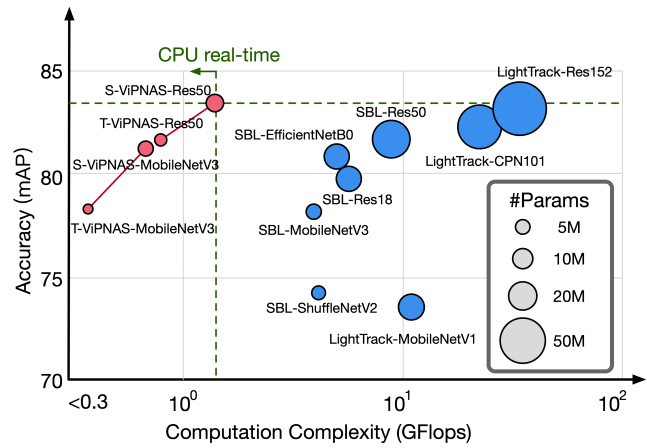For video pose estimation, there is commonly consider-

Figure 1. **Speed-accuracy trade-off** on PoseTrack2018 [1] validation set. Methods involve SBL [56], LightTrack [38] and our ViPNAS with various backbones. With accuracy comparable to state-of-the-art networks, ViPNAS achieves CPU real-time with significantly lower computation.

able temporal redundancy that leads to superfluous computation, *i.e.* adjacent frames in a video share similar global context information. The temporal contextual information can be used for improving pose estimation. Therefore, it is critical to fuse features from adjacent frames to the current frame in order to effectively utilize the temporal contextual information for balancing accuracy and efficiency. However, there are still several open questions:

1. Low-level local features are important for accurate localization, while higher-level global features are robust to occlusion and large pose variations. Which stage of features should be fused?

2. For temporal feature fusion, various fusion operations (e.g. addition, multiplication, or concatenation) are chosen by trial-and-error. How to choose the optimal operation?

3. The goal is to optimize the total accuracy subject to the total computation complexity (Flops) constraints over

the whole video. Previous works generally explicitly enforce different frames to apply the same model, which will result in sub-optimal performance. How to efficiently allocate computation across different video frames?

Manually exploring the design choices regarding the above questions via trial-and-error can be tedious. We instead apply neural architecture search (NAS) to give a unified solution to them. We propose a novel spatial-temporal NAS framework for efficient video pose estimation, termed ViPNAS. For spatial-level search, we optimize the neural architecture by a wide spectrum of five dimensions (depth, width, kernel size, group number, and attention). For temporal-level search, we jointly search three aspects of designs: 1) the stage of features to be fused, 2) the feature fusion operation, and 3) the allocation of computation across video frames. The spatial-level and temporal-level search are jointly optimized through a single framework. Given the total Flops over multiple frames as constraints, we can efficiently allocate computation across different video frames for optimizing performance. Experiments show that ViP-NAS significantly improves over the state-of-the-art methods, such as SBL [56] and LightTrack [38], with various well-known backbones (ResNet [13], CPN [6], MobileNets [15, 14], ShuffleNet [35] and EfficientNet [51]).

Our main contributions can be summarized as follows:

- We propose the novel spatial-temporal neural architecture search (NAS) framework for efficient video pose estimation, termed ViPNAS.

- ViPNAS learns to allocate computational resources (*e.g.* Flops) for different frames under the total computation complexity constraints across frames.

- ViPNAS automatically searches temporal connections, *i.e.* the fusion module and positions. In the task of video pose estimation, we achieve the state-of-the-art accuracy with CPU real-time performance (> 25 FPS).

2. Related Work

2.1. Human Pose Estimation

Recent works in human pose estimation [6, 7, 9, 19, 21, 24, 33, 36, 49, 54, 56] focus on designing stronger neural network architectures with higher model capacity to improve accuracy. To better capture the context information, the attention mechanism has been successfully applied in human pose estimation. For example, Chu *et al*. [8] proposes multi-context attention to improve model robustness and accuracy. Su *et al*. [48] proposes SCARB module to enhance pyramid features via spatial and channel-wise context. Other popular attention modules have also been widely explored. For example, Squeeze-and-Excitation (SE) block [16] models channel-wise relationship and Global Context (GC) block [4] models the global

context via addition fusion as NLNet [53]. Different from manually design in these works, we propose to apply NAS to automatically search for optimal architectures.

For online video pose estimation, some works [18, 20, 49, 56, 57, 58] directly apply the image-based pose models on each video frame. However, such approaches do not capture the temporal consistency, suffering from motion blur or occlusion. Other works utilize temporal cues in order to keep geometric consistency across frames. Such approaches include directly processing concatenated consecutive frames along the channel-axis [42], applying 3D temporal convolution [10, 52], using dense optical flow to produce smooth movement [41, 47]. These models are typically computationally expensive, making them not applicable in real-time applications. Recently, some works [11, 23, 25, 34, 37] follow the pose propagation paradigm, that transfer features from previous frames to the current frame in an online fashion. However, how to choose the temporal feature fusion sites and the fusion operations are still open questions. We aim to answer this by applying the ViPNAS framework to explore the most effective combination.

2.2. Neural Architecture Search

NAS for image-level tasks. Neural architecture search (NAS) focuses on automating the neural network architecture design. Early NAS approaches [29, 45, 50, 64, 65] sample a large number of architectures and trained them from scratch, which are very time consuming. Recent NAS approaches [2, 3, 26, 27, 30, 31, 32, 55, 61, 63] adopt a weight sharing strategy and train the super-network. Our method also follows this paradigm that trains the super-network only once, and evaluates various sub-networks.

NAS for video-level tasks. NAS has been applied in video-level tasks, such as video recognition [40, 43, 44, 46]. EVANet [44] searches for sequential or parallel model configurations via evolutionary algorithm. AssembleNet [46] searches for multi-stream (RGB and optical flow) network connectivity. TinyVideoNet [43] searches for computationally efficient classification model for video recognition.

NAS for single-image pose estimation. PoseNFS [59] introduces the prior structure of the human body and searches for multiple personalized modules for part-based representations. AutoPose [12] proposes a bi-level optimization method that combines reinforcement learning and gradient-based method.

Our work is different from existing works on NAS in three aspects. First, we are the first to apply NAS for a challenging task of video pose estimation. Second, existing works for image-level and video-level tasks do not search for different architectures at once, but our work search frame-specialized models for further leveraging the ability of NAS in video pose estimation. Third, we propose the novel spatial and temporal search space for the task.

To fully exploit the temporal information, we search for the optimal combination when fusing features from previous frames to the current frame, which was not explored in these works. Our method also inherits the merit of once-for-all [2], *i.e.* training only once and obtaining many sub-networks, which effectively reduces the searching cost.

3. Method

3.1. Overview

Video pose estimation aims to localize the human body parts (also referred to as keypoints or joints) of a person instance in each frame.

In this paper, based on the online pose propagation paradigm, we propose a novel NAS framework for efficient video pose estimation (ViPNAS). The pipeline of ViPNAS is shown in Figure 2. The first frame is selected as the key frame for every $T + 1$ frames in the video. For a key frame, a high-accuracy spatial video pose estimation network (S-ViPNet) is applied to localize human poses. We follow the common settings to use heatmaps to encode the joint locations as Gaussian peaks. For a non-key frame, a lightweight temporal video pose estimation network (T-ViPNet) is used for pose propagation. In T-ViPNet, some CNN layers are used for extracting the features of the current frame, then a temporal feature fusion module fuses the features of the current frame and the heatmaps of the last frame. The fused features are then processed by the remaining CNN layers of the T-ViPNet to obtain the heatmaps. The predicted heatmaps encode the per-pixel likelihood of each joint, which are informative cues to guide the keypoint localization in the subsequent frames. The propagation continues until the next key frame.

ViPNAS contains two levels of search space, *i.e.* the spatial-level and the temporal-level. The architecture of the key frame (S-ViPNet) is searched in the spatial-level search space. The architectures of the non-key frames (T-ViPNets), including the temporal feature fusion module and CNN layers, are searched in both spatial-level and temporal-level search space. Different non-key frames have different architectures in both the feature fusion module (fusion operation and feature fusion stage) and CNN layers, as shown by the example for frames $t + 1$ and $t + 2$ in Figure 2.

3.2. Spatial-level Search Space

Motivated by [2, 61], we design the weight shared super-network for model architecture search and search for the block number and block structure. Our architecture search spaces extend [2, 61] to include group and attention for a wider spectrum of five dimensions (depth, width, kernel size, group, and attention). We find out the best configuration of these settings. Our super-network is divided into several stages in series and each stage consists of several blocks having the same spatial resolution of output features. We search on five dimensions as follows:

Elastic Depth: The number of blocks for each stage. We activate the first D blocks of a stage when the depth D is selected for this stage. **Elastic Width**: The number of output channels in each block. We keep the first W filters when the width W is selected. **Elastic Kernel Size**: The kernel size of convolutional layers in each block. We reserve the centering $K \times K$ convolutional kernel when the kernel size K is selected. The possible choices of kernel size K are $\{3, 5, 7\}$ for normal convolutional layers and $\{2, 4\}$ for deconvolutional layers. **Elastic Group Number**: The group number of convolutional layers [22] in each block. It ranges from 1 (standard convolution) to N (depth-wise convolution) for N input channels. **Elastic Attention Module**: Using the attention module or not at the end of each block. Since attention modules are shown to be effective for pose estimation in [8, 48], we include attention modules in our search space. We investigate whether to use the attention module (*e.g.* GC block [4] or SE Block [16]) at the end of each block. If the attention module is not selected, we skip the attention module and identity mapping is applied. Please refer to Sec. A.1 for more details about the spatial-level search space.

3.3. Temporal-level Search Space

Lightweight pose models alone have difficulty in capturing the global information and distinguishing the joints with similar appearance. However, considering that poses in adjacent video frames are temporally correlated, lightweight models can estimate the joint locations with the local appearance and the guidance from previous frames.

Temporal feature fusion is critical to the task of video pose estimation, which has also been explored in literature [11, 23, 25, 34, 37]. Previous works on temporal fusion mainly differ in two main aspects, *i.e.* the fusion operations and the feature fusion stages. Popular fusion operations may include addition (Add), multiplication (Mul), and concatenation (Cat), *etc*. As different pose networks prefer different fusion operations, the choice of the fusion operation is carefully hand-crafted. Besides, different stages of the input features are fused in different approaches. Generally, low-level features may contain more detailed localization information, while higher-level features may contain more global information. In previous works, the levels of features used are mainly chosen by trial-and-error. In ViPNAS, we instead allow the networks to automatically search for the optimal fusion operation and the best stage of features to fuse in a single run of the search.

As shown in Figure 2, our designed temporal feature fusion module includes two inputs, *i.e.* heatmaps of the previous frame and features of the current frame $t + 1$. The temporal feature fusion module first selects the location of
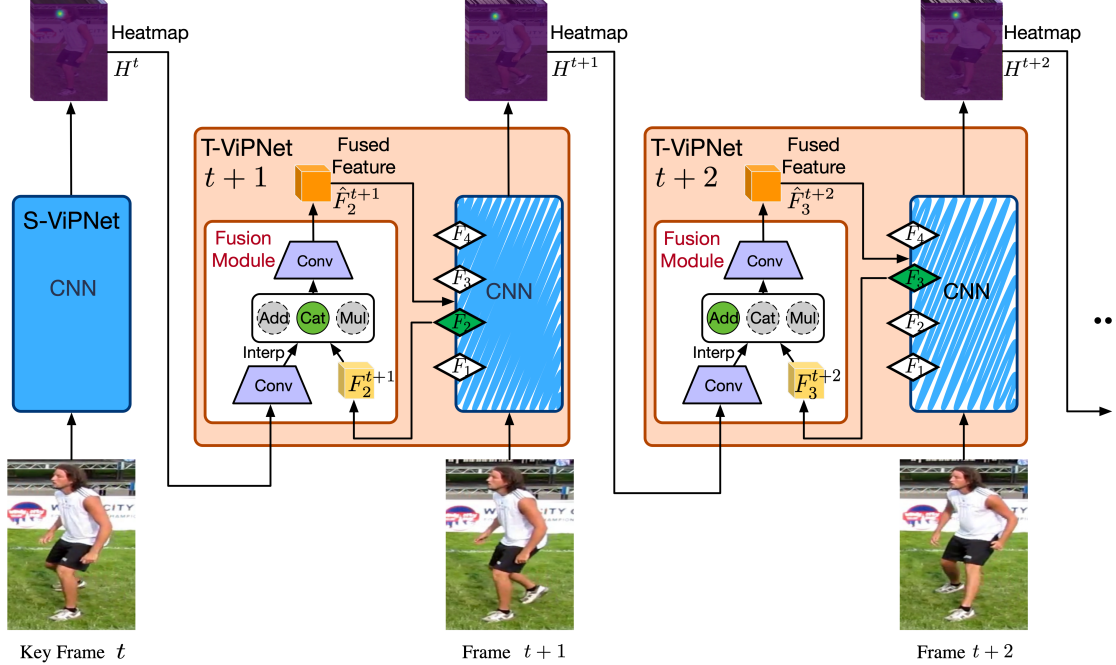
Figure 2. ViPNAS consists of one image-based key frame pose model S-ViPNet, and T video-based pose model T-ViPNets containing temporal feature module and various CNN architectures. Videos are processed frame-by-frame in an online mode. S-ViPNet first predicts the pose heatmaps H^t of the key frame t, and propagates them to the next frame $t+1$. T-ViPNet selects the CNN architecture, as well as the input features (*e.g.* F_1 to F_4) and fusion operation (*e.g.* Add, Cat and Mul) of fusion module. The fusion module combines the selected feature F_2^{t+1} with the propagated heatmaps H^t, and generates the fused features $\hat{F}_2^{t+1}$ for predicting the heatmaps H^{t+1}.

the input features F_2^{t+1}. The pose heatmaps from the adjacent frame are processed by one 1×1 convolution, followed by one bi-linear interpolation layer to adjust the channels and the resolution (width & height) to match those of the *selected* features F_2^{t+1}. The heatmaps and features are then fused by the *selected* fusion operator (Cat), which are then processed by one 1×1 convolution, making the fused features ($\hat{F}_2^{t+1}$) have the same shape as the input features.

Our temporal-level search space for a non-key frame includes N_O choices for the feature fusion operations, *e.g.* addition (Add), multiplication (Mul), and concatenation (Cat); and N_S choices for the input feature stages, *e.g.* F_1, F_2, F_3, and F_4. The size of temporal search space is $(N_O \times N_S)^T$, which is impossible to optimize by trial-and-error.

3.4. Train and Search for ViPNAS

3.4.1 Train and Search for S-ViPNet

Based on the spatial-level search space defined in Section 3.2, we use the approach in [61] to train the super-network. Sandwich rule [60, 61] and in-place distillation [60, 61] are applied. Then we sample the sub-networks under the given constraint and evaluate each of them on the validation set to search the architecture of S-ViPNet, which is the network for the key frame.

3.4.2 Training for T-ViPNet

In this section, we introduce the multi-frame propagation training scheme of our ViPNAS. The goal is to optimize the overall model accuracy at spatial and temporal levels simultaneously in the process of poses propagation across multiple video frames. The overall objective function can be formulated as follows:

$$\min_{\theta_\mathcal{T}} \sum_{t=1}^{T} \sum_{\mathrm{arch}^t} \mathcal{L}(\mathcal{T}(I^t, H^{t-1}; \{\theta_\mathcal{T}, \mathrm{arch}^t\})), \quad (1)$$

$$\text{where } H^t = \begin{cases} \mathcal{T}(I^t, H^{t-1}; \{\theta_\mathcal{T}, \mathrm{arch}^t\}), & t \geq 1, \\ \mathcal{S}(I^t; \{\theta_\mathcal{S}\}), & t = 0. \end{cases} \quad (2)$$

$\mathcal{S}$ is the key frame model S-ViPNet, whose weights are denoted by $\theta_\mathcal{S}$. It is pre-trained and fixed when training and searching for T-ViPNets. $\mathcal{T}$ is the super-network of T-ViPNets, which is parameterized by $\theta_\mathcal{T}$. During training, we sample sub-network consisting of architecture arch^t from $\mathcal{T}$ and the weights of this architecture copied from the super-network weights $\theta_\mathcal{T}$. For each frame t, I^t is the input image, and H^t is the predicted heatmaps. We use MSE loss function $\mathcal{L}$ to measure the difference between the target heatmaps and the predicted ones of each non-key frame.

The multi-frame pose propagation training of T-ViPNet is shown in Figure 3, where the heatmaps of the key frame are propagated to T ($T \geq 2$) non-key frames iteratively. We apply a single super-network $\mathcal{T}$ for all the non-key frames and all T-ViPNets share the weights, which saves memory during training. Moreover, with one-time training of the super-network, we can search for multiple sets of T-ViPNets with various numbers of propagation frames, see Table. 4. We make the super-network $\mathcal{T}$ to share the same CNN architecture as the discovered S-ViPNet. First, since the tasks of image-based and video-based pose estimation are highly correlated, the good-performing image-based pose estimator can serve as a good candidate architecture for video pose estimation. Second, the pre-trained weights of S-ViPNet can be reloaded for the initialization of the super-network. Third, by sharing similar architectures, the features for the key frame and non-key frames are better aligned.

We jointly train the temporal models (T-ViPNets) in the spatial-level and temporal-level search space for the global optimum. We apply the *Sandwich rule* [60, 61] to sample the smallest sub-network, the biggest sub-network and N randomly sampled sub-networks (N = 2 in our experiments) for each mini-batch. We train and search for the CNN architectures and temporal fusion module (including fusion operations and fusion stages) simultaneously. For the biggest (or smallest) sub-network, T-ViPNets of all the frames use the biggest (or smallest) CNN architectures, while the temporal-level search spaces are randomly sampled. For N randomly sampled sub-networks, each T-ViPNet samples unique architecture at both spatial and temporal search spaces. *Inplace knowledge distillation* [60, 61] takes the prediction of biggest sub-network as the soft labels to enhance supervision for other sub-networks. The biggest sub-network is supervised by ground truth heatmaps with MSE loss, while others are supervised by both the soft labels and the ground truth heatmaps with equal loss weights.

3.4.3 Automatic Computation Allocation

As stated above, the sub-networks (T-ViPNets) of different frames do not necessarily share the same architecture. In ViPNAS, different model complexities are assigned to different frames automatically.

Formally, we aim to search for a group of sub-network architectures ($\{\text{arch}^t\}_{t=1:T}$) that optimize the overall Average Precision (AP) under the overall computation complexity (Flops) constraints C:

$$\max_{\text{arch}^{1:T}} \sum_{t=1}^{T} \text{AP}(\mathcal{T}(I^t, H^{t-1}; \{\theta_{\mathcal{T}}, \text{arch}^t\}))$$
$$\text{s.t.} \quad \sum_{t=1}^{T} \text{Flops}(\text{arch}^t) \leq \text{C}$$
(3)
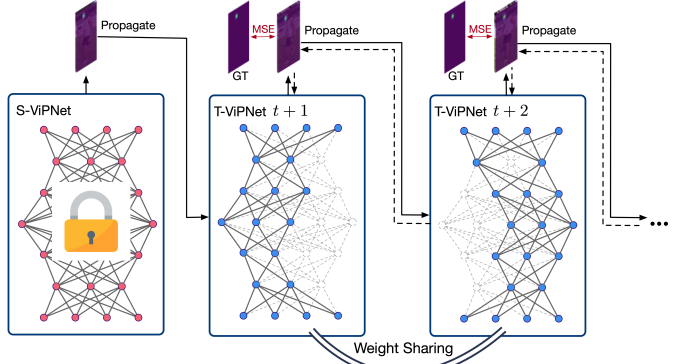
In the search process, we simply follow [61] to randomly

Figure 3. Multi-frame pose propagation training of ViPNAS. The key frame S-ViPNet is first pre-trained and fixed. T ($T \geq 2$) various non-keyframe sub-networks are sampled from a single super-network with sharing weights, and jointly supervised by MSE loss for each frame. The solid lines indicate the forward process and the dotted lines indicate the back propagation process.

sample sub-networks that fulfill the given constraints and evaluate the accuracy on the validation set. The sampled sub-networks with the highest AP on the validation set under the Flops constraint are used as the T-ViPNets.

4. Experiments

4.1. Datasets

COCO2017 Dataset [28] is a standard benchmark for human pose estimation. It contains over 200,000 images and 250,000 person instances. We train the models on the COCO train2017 dataset (57K images), and evaluate them on the val2017 set (5K images) and test-dev2017 set (20K images) using the official evaluation metric[1]: Average Precision (AP) and Average Recall (AR), which are based on the standard object keypoints similarity (OKS). OKS $= \frac{\sum_i \exp(-d_i^2/2s^2k_i^2)\delta(v_i>0)}{\sum_i \delta(v_i>0)}$, where d_i is the Euclidean distance between each ground-truth and the detected keypoint, v_i is the visibility flag, s is the scale of person, and k_i is a constant to control falloff.

PoseTrack2018 Dataset [1] is a large-scale dataset for human pose estimation in videos. It contains various videos of human activities with 6 person instances per frame on average. We use PoseTrack2018 V0.25 annotation, which includes 593 training videos, 74 validation videos and 375 testing videos. We follow the common settings [38, 49, 56] to pre-train models on COCO train2017 dataset and fine-tune them on PoseTrack2018 training set. The evaluation follows [23, 25, 34, 37] for video pose estimation [17, 62] that estimates human poses given ground-truth bounding boxes. Pose estimation accuracy is evaluated using the standard AP metric[2].

---

[1] http://cocodataset.org/#keypoints-eval
[2] https://posetrack.net/

4.2. Implementation Details

We train and search our single-frame pose estimator, termed S-ViPNAS, on COCO dataset. For training, we resize the cropped person image to 256×192, and apply random rotation ($[-40°, 40°]$) and random flip as data augmentation. We train the super-network with inplace knowledge distillation for 250 epochs. Weights are initialized from zero-mean Gaussian distribution with $\sigma = 0.001$. The basic learning rate is 1e-3, and is reduced by a factor of 10 at the 200th and 230th epoch. We sample 500 models under the Flops constraints and search for S-ViPNAS with the highest AP on the validation set.

We directly transfer the discovered architecture (S-ViPNAS) on the COCO dataset to the PoseTrack dataset. We fine-tune S-ViPNAS on PoseTrack dataset for 20 epochs. The basic learning rate is 1e-4, and drops to 1e-5 at 10 epochs then 1e-6 at 15 epochs. We use S-ViPNAS as the key frame pose estimator and the super-network for temporal propagation models (T-ViPNAS). During multi-frame super-network training, the same augmentation methods are applied across $T+1$ frames ($T = 3$ by default). We train the super-network using the sandwich rule for 60 epochs with initial learning rate 1e-3 and cosine learning rate schedule. The search cost is 16 GPU days for training and 2GPU days for search on V100 GPUs.

4.3. ViPNAS for efficient video pose estimation

Table 1 compares our proposed ViPNAS with the state-of-the-art methods on PoseTrack2018 [1] validation set.

SBL [56] proposes to add deconvolutional layers to the backbone network, which has been proved effective. We extend [56] to include more well-known efficient backbones for comparisons, such as EfficientNet [51], ShuffleNet [35], and MobileNet [14]. These models are pre-trained on COCO dataset and fine-tuned on PoseTrack dataset with the same experimental configurations as [56]. LightTrack [38] is a recently proposed light-weight framework for video pose estimation. The results are obtained using the official codes[3] with the released pre-trained models.

We evaluate our methods on two well-known backbones, *i.e.* ResNet-50 [13] and MobileNet-V3 [14]. For both backbone models, we build the super-network based on the spatial-level search space (Sec. 3.2) and temporal-level search space (Sec. 3.3). Please refer to Sec. A.2 for more details. SBL, LightTrack, and S-ViPNAS directly apply the image-based pose models on each video frame, while T-ViPNAS searches for temporal feature fusion for more efficient pose estimation. #Param and Flops are calculated by averaging over the whole video frames including both key frames and non-key frames. From Table 1, we see that ViPNAS achieves the state-of-the-art accuracy with signif-
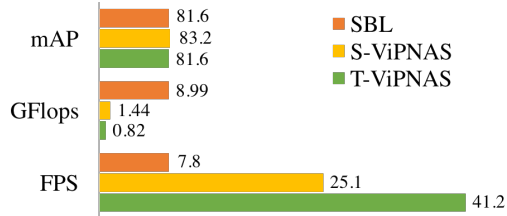
---

[3]https://github.com/Guanghan/lighttrack

Figure 4. **Comparisons** among SBL [56], S-ViPNAS and T-ViPNAS with ResNet-50 backbone. ViPNAS discovers models with much less computational complexity and significantly higher speed (single core of a 3.2GHz Intel i7-8700 CPU).

icantly lower model complexity. T-ViPNAS significantly boosts the model efficiency and reduces the computation without sacrificing the overall accuracy. For example, T-ViPNAS-MobileNetV3 achieves 10x Flops reduction (0.37 vs 4.1) without accuracy drop (78.2 vs 78.1).

Figure 4 compares SBL [56], S-ViPNAS and T-ViPNAS with ResNet-50 backbone on PoseTrack2018 validation set. We report mAP, GFlops, and speed (FPS). Speed is evaluated on a single core of an Intel i7-8700 CPU (3.2GHz). We show that T-ViPNAS is significantly faster (41FPS on CPU) than the baseline, with comparable accuracy, making it practical for real-world applications.

4.4. ViPNAS for image-based pose estimation

Table 2 demonstrates the performance of the discovered S-ViPNAS models on COCO2017 dataset, compared with other state-of-the-art hand-crafted methods and concurrent NAS based pose estimators. We report our discovered results based on multiple backbones (*i.e.* HRNet-W32 [49], ResNet-50 [13] and MobileNetV3 [14]). For fair comparisons, we retrain S-ViPNAS models using the same training recipe and use the same Faster-RCNN human detection bounding boxes as SBL [56] and HRNet [49].

We see that our discovered S-ViPNAS-HRNetW32 significantly outperforms the popular hand-crafted models and the NAS based models. Compared with the current state-of-the-art HRNet [49], we achieve higher accuracy and lower complexity (5.64 vs 7.10 GFlops). Compared with other NAS pose models PoseNFS [59] and AutoPose [12], ViP-NAS also shows superiority in terms of both accuracy and computation complexity. Note that AutoPose [12] uses a stronger human detector [5] on COCO val2017 set.

We further search for lightweight pose estimators to boost the model efficiency. Based on ResNet-50 [56], we obtain a 6x smaller (1.44 vs 8.90 GFlops) model (S-ViPNAS-Res50) without sacrificing the accuracy. For MobileNet-V3 [14], our method finds a 5.8x smaller (0.69 vs 4.06 GFlops) model (S-ViPNAS-MobileNet) with 3.1mAP gain (67.8 vs 64.7) on COCO val2017 set.

Table 1. **Comparisons** with other video pose estimation approaches on PoseTrack2018 validation set. Our ViPNAS achieves the state-of-the-art performance with significantly lower computation complexity.

Method	Backbone	Image Size	#Params	GFLOPs	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Total AP
SBL [56]	ShuffleNetV2 [35]	256×192	4.7M	4.21	84.5	83.0	74.4	63.1	74.4	70.7	63.4	63.4
SBL [56]	EfficientNetB0 [51]	256×192	14.9M	5.05	87.3	87.9	82.0	73.4	79.0	79.2	72.9	80.7
SBL [56]	ResNet-18 [13]	256×192	15.3M	5.79	86.5	86.9	80.8	71.5	79.3	77.9	70.6	79.6
LightTrack [39]	MobileNetV1 [15]	384×288	15.8M	11.3	85.2	81.7	74.7	62.9	72.9	69.4	61.4	73.4
LightTrack [39]	CPN101 [6]	384×288	46.3M	22.9	87.9	87.7	83.5	75.8	79.1	80.4	77.0	82.1
LightTrack [39]	ResNet152 [13]	384×288	68.6M	35.6	89.4	88.5	84.4	76.2	81.2	80.5	77.5	83.0
SBL [56]	MobileNet-V3 [14]	256×192	5.5M	4.1	86.4	85.9	78.8	69.6	76.5	76.1	69.1	78.1
S-ViPNAS	MobileNet-V3 [14]	256×192	5.4M	0.69	87.8	88.0	82.3	74.1	78.8	79.1	74.0	81.1
T-ViPNAS	MobileNet-V3 [14]	256×192	2.5M	**0.37**	87.3	85.6	78.9	70.3	75.7	75.0	70.1	78.2
SBL [56]	ResNet-50 [13]	256×192	34.0M	8.99	86.7	88.1	83.0	75.7	80.8	80.4	74.2	81.6
S-ViPNAS	ResNet-50 [13]	256×192	7.3M	1.44	88.1	89.6	84.5	77.4	81.1	81.8	77.6	**83.2**
T-ViPNAS	ResNet-50 [13]	256×192	3.9M	0.82	87.7	88.2	82.6	74.7	79.3	79.8	75.4	81.6

Table 2. **Comparisons** on COCO2017 dataset. Our approach significantly outperforms other hand-crafted and NAS models in terms of both speed and accuracy on COCO val2017 set and test-dev2017 set. [†] means using a stronger person bounding box detector (HTC [5]).

	Method	Image Size	#Params	GFLOPs	AP	AP^{50}	AP^{75}	AP^M	AP^L	AR
				COCO Val2017 Set						
Hand-Crafted Models	MobileNet-V3[14]	256×192	5.5M	4.06	64.7	86.7	72.6	61.4	70.9	76.3
	SBL-50 [56]	256×192	34.0M	8.90	70.4	88.6	78.3	67.1	77.2	76.3
	HRNet-W32[49]	256×192	28.5M	7.10	74.4	90.5	81.9	70.8	81.0	79.8
NAS Models	PoseNFS-3 [59]	384×288	6.1M	4.0	68.0	-	-	-	-	-
	PoseNFS-3 [59]	384×288	15.8M	14.8	73.0	-	-	-	-	-
	AutoPose [12][†]	256×192	-	10.65	73.6	90.6	80.1	69.8	79.7	78.1
Ours	S-ViPNAS-MobileV3	256×192	**2.8M**	**0.69**	67.8	87.2	76.0	64.7	74.0	75.2
	S-ViPNAS-Res50	256×192	13.5M	1.44	71.0	89.3	78.7	67.7	77.5	76.7
	S-ViPNAS-HRNetW32	256×192	16.3M	5.64	**74.7**	89.9	82.0	71.0	81.5	81.2
				COCO Test-Dev2017 Set						
Hand-Crafted Models	SBL-50 [56]	256×192	34.0M	8.90	70.0	90.9	77.9	66.8	75.8	75.6
	HRNet-W32[49]	256×192	28.5M	7.10	73.5	91.6	81.7	70.1	79.1	80.1
NAS Models	PoseNFS-3 [59]	384×288	6.1M	4.0	67.4	89.0	73.7	63.3	74.3	73.1
	PoseNFS-3 [59]	384×288	15.8M	14.8	72.3	90.9	79.5	68.4	79.2	77.9
Ours	S-ViPNAS-Res50	256×192	**13.5M**	**1.44**	70.3	90.7	78.8	67.3	75.5	77.3
	S-ViPNAS-HRNetW32	256×192	16.3M	5.64	**73.9**	91.7	82.0	70.5	79.5	80.4

Table 3. Effect of temporal-level NAS for video pose estimation on PoseTrack2018 dataset. Given the same GFlops constraints, T-ViPNAS discovers better architectures with higher accuracy.

Method	Backbone	GFLOPs	mAP
S-ViPNAS-a	ResNet-50	0.82	80.3
T-ViPNAS-a	ResNet-50	0.82	**81.6**
S-ViPNAS-b	MobileNet-V3	0.37	77.2
T-ViPNAS-b	MobileNet-V3	0.37	**78.2**

4.5. Ablation Study

Effect of temporal-level search. To validate the effect of temporal-level search, we search S-ViPNAS under the constraints of the same model complexity as T-ViPNAS. We apply the image-based S-ViPNAS models independently for each frame. As shown in Table 3, we see that given the same Flops constraints, T-ViPNAS discovers better model architectures with higher accuracy (81.6 vs 80.3 mAP for ResNet-50 based models (-a) and 78.2 vs 77.2 mAP for MobileNet-V3 based models (-b)).

Effect of temporal feature fusion. As shown in Figure 6(a), we explore the effect of temporal feature fusion

on the PoseTrack2018 validation set. We search for four groups of T-ViPNAS models with ResNet-50 backbone in a range of average computation complexity levels (from 0.8 to 1.2 GFLOPs) for comparisons. The number of propagation frames is set as $T = 3$, so for each group, we have 4 different models (*i.e.* 1 S-ViPNet and 3 T-ViPNets) in total.

To validate the effectiveness of temporal feature fusion, we remove the temporal feature fusion from T-ViPNAS (red) in each group, keep the model architecture the same, and re-train them based on single images (blue). We see that our T-ViPNAS consistently improves over the baselines for various Flops requirements. Our experiments show that temporal fusion captures the consistency among adjacent frames and propagates poses efficiently using extremely lightweight models.

Effect of automatic computation allocation. As shown in Figure 6(b), we further explore the effect of automatic computation allocation on the PoseTrack2018 validation set. For comparisons, we search for the temporal models sharing both the spatial and temporal architectures (blue) under the same Flops constraints as our T-ViPNAS (red).
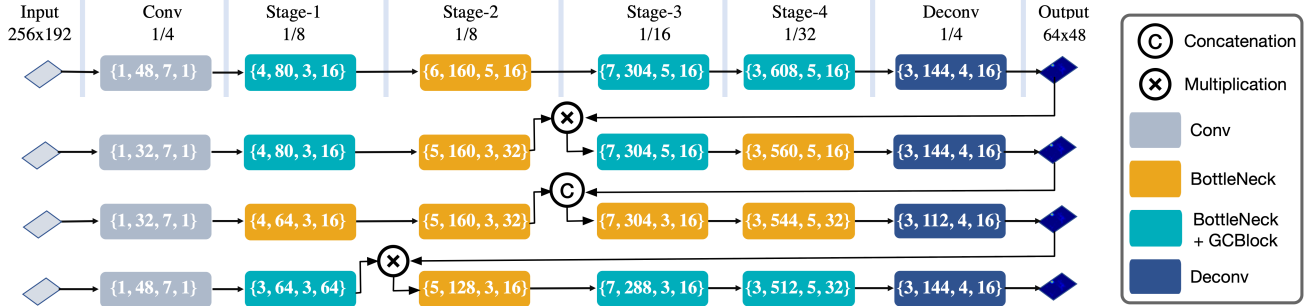
Figure 5. Example of T-ViPNAS with ResNet-50 backbone. {Depth, Width, Kernel Size, Group} are listed in the figure.

(a) Ablation study on searching for temporal feature fusion

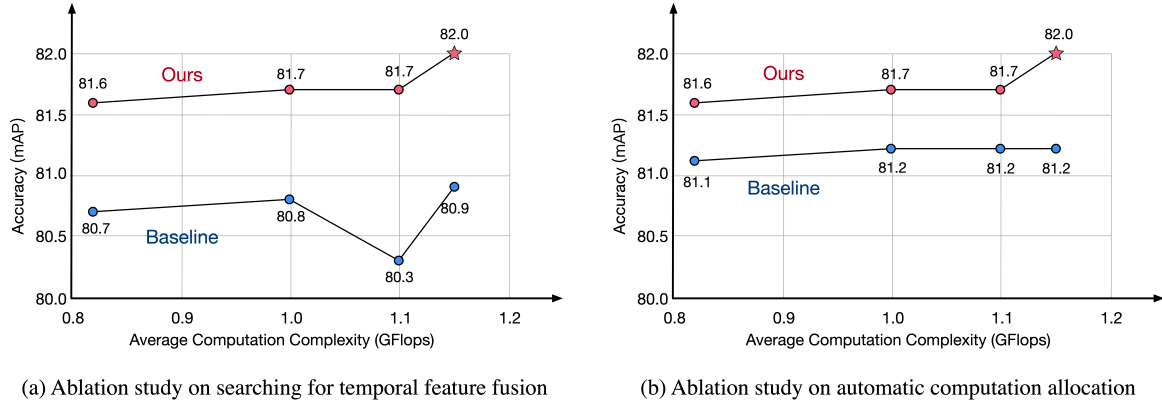(b) Ablation study on automatic computation allocation

Figure 6. Comparing T-ViPNAS with (a) baselines without temporal feature fusion modules (b) baselines with the same architectures for different frames. We see that our proposed T-ViPNAS consistently improves over the baseline architectures for a range of complexity levels (from 0.8 to 1.2 GFlops). We visualize the architecture of one example T-ViPNAS (red star) in Figure 5.

We find that our T-ViPNAS consistently improves over the baseline architectures by at least 0.5% mAP, demonstrating the effectiveness of automatic computation allocation that searches for frame-specialized models. Example architectures of our discovered models are visualized in Figure 5.

Effect of the number of propagation frames. We evaluate the transferability of our proposed ViPNAS training scheme to various propagation frames T. During training of T-ViPNAS (Sec. 3.4.2) with ResNet-50 backbone, we set the number of non-key frames as $T = 3$, but search on different propagation lengths without re-training the super-network. As shown in Table 4, we set the constraints of the average model computation complexity to be 1.0 GFlops, and search for different propagation frame numbers, $i.e.$ $T = 2, T = 3$ or $T = 4$ frames. We see that our ViPNAS is relatively robust to the number of propagation frames.

5. Conclusion

In this paper, we propose ViPNAS for online video pose estimation, trading-off between accuracy and the computation cost. ViPNAS automatically allocates computation resources ($i.e.$ Flops) for different frames to achieve the overall optimum. By designing the novel spatial-temporal

Table 4. Effect of the number of propagation frames. We experiment with training the super-network for $T = 3$ frames and searching for $T = \{2, 3, 4\}$ frames with 1.0 GFlops complexity constraint on average.

#Propagation Frames (T)	GFLOPs	mAP
2	1.0	81.7
3	1.0	81.7
4	1.0	81.4

search space, we can simultaneously search for CNN architectures and temporal connections, $i.e.$ the fusion operations and the feature fusion sites. Empirical experiments demonstrate that our proposed ViPNAS successfully discovers the architecture that achieves the state-of-the-art accuracy with CPU real-time performance.

References

[1] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *Int. Conf. Learn. Represent.*, 2020.

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *Int. Conf. Learn. Represent.*, 2019.

[4] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Int. Conf. Comput. Vis. Worksh.*, 2019.

[5] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[6] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[7] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[8] Xiao Chu, Wei Yang, Wanli Ouyang, Cheng Ma, Alan L Yuille, and Xiaogang Wang. Multi-context attention for human pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[9] Haodong Duan, Kwan-Yee Lin, Sheng Jin, Wentao Liu, Chen Qian, and Wanli Ouyang. Trb: a novel triplet representation for understanding 2d human body. In *Int. Conf. Comput. Vis.*, 2019.

[10] Rohit Girdhar, Georgia Gkioxari, Lorenzo Torresani, Manohar Paluri, and Du Tran. Detect-and-track: Efficient pose estimation in videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[11] Georgia Gkioxari, Alexander Toshev, and Navdeep Jaitly. Chained predictions using convolutional neural networks. In *Eur. Conf. Comput. Vis.*, 2016.

[12] Xinyu Gong, Wuyang Chen, Yifan Jiang, Ye Yuan, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Autopose: Searching multi-scale branch aggregation for pose estimation. *arXiv preprint arXiv:2008.07018*, 2020.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[14] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Int. Conf. Comput. Vis.*, 2019.

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

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[17] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *Eur. Conf. Comput. Vis.*, 2013.

[18] Sheng Jin, Wentao Liu, Wanli Ouyang, and Chen Qian. Multi-person articulated tracking with spatial and temporal embeddings. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[19] Sheng Jin, Wentao Liu, Enze Xie, Wenhai Wang, Chen Qian, Wanli Ouyang, and Ping Luo. Differentiable hierarchical graph grouping for multi-person pose estimation. In *Eur. Conf. Comput. Vis.*, 2020.

[20] Sheng Jin, Xujie Ma, Zhipeng Han, Yue Wu, Wei Yang, Wentao Liu, Chen Qian, and Wanli Ouyang. Towards multi-person pose tracking: Bottom-up and top-down methods. In *ICCVW*, 2017.

[21] Sheng Jin, Lumin Xu, Jin Xu, Can Wang, Wentao Liu, Chen Qian, Wanli Ouyang, and Ping Luo. Whole-body human pose estimation in the wild. In *Eur. Conf. Comput. Vis.*, 2020.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 2017.

[23] Haihan Li, Wenming Yang, and Qingmin Liao. Temporal feature enhancing network for human pose estimation in videos. In *IEEE Int. Conf. Image Process.* IEEE, 2019.

[24] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[25] Wentian Li, Xiangyu Xu, and Yu-Jin Zhang. Temporal feature correlation for human pose estimation in videos. In *IEEE Int. Conf. Image Process.* IEEE, 2019.

[26] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[27] Feng Liang, Chen Lin, Ronghao Guo, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Computation reallocation for object detection. In *Int. Conf. Learn. Represent.*, 2019.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Eur. Conf. Comput. Vis.*, 2014.

[29] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Eur. Conf. Comput. Vis.*, 2018.

[30] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *Int. Conf. Learn. Represent.*, 2019.

[31] Jie Liu, Chuming Li, Feng Liang, Chen Lin, Ming Sun, Junjie Yan, Wanli Ouyang, and Dong Xu. Inception convolution with efficient dilation search. *arXiv preprint arXiv:2012.13587*, 2020.

[32] Jiaheng Liu, Shunfeng Zhou, Yichao Wu, Ken Chen, Wanli Ouyang, and Dong Xu. Block proposal neural architecture search. *IEEE Trans. Image Process.*, 2020.

[33] Wentao Liu, Jie Chen, Cheng Li, Chen Qian, Xiao Chu, and Xiaolin Hu. A cascaded inception of inception network with attention modulated feature fusion for human pose estimation. In *AAAI*, 2018.

[34] Yue Luo, Jimmy Ren, Zhouxia Wang, Wenxiu Sun, Jinshan Pan, Jianbo Liu, Jiahao Pang, and Liang Lin. Lstm pose machines. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[35] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Eur. Conf. Comput. Vis.*, 2018.

[36] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Eur. Conf. Comput. Vis.*, 2016.

[37] Xuecheng Nie, Yuncheng Li, Linjie Luo, Ning Zhang, and Jiashi Feng. Dynamic kernel distillation for efficient pose estimation in videos. In *Int. Conf. Comput. Vis.*, 2019.

[38] Guanghan Ning, Jian Pei, and Heng Huang. Lighttrack: A generic framework for online top-down human pose tracking. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2020.

[39] Guanghan Ning, Jian Pei, and Heng Huang. Lighttrack: A generic framework for online top-down human pose tracking. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2020.

[40] Wei Peng, Xiaopeng Hong, and Guoying Zhao. Video action recognition via neural architecture searching. In *IEEE Int. Conf. Image Process.*, 2019.

[41] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing convnets for human pose estimation in videos. In *Int. Conf. Comput. Vis.*, 2015.

[42] Tomas Pfister, Karen Simonyan, James Charles, and Andrew Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In *ACCV*, 2014.

[43] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Tiny video networks. *arXiv preprint arXiv:1910.06961*, 2019.

[44] AJ Piergiovanni, Anelia Angelova, Alexander Toshev, and Michael S Ryoo. Evolving space-time neural architectures for videos. In *Int. Conf. Comput. Vis.*, 2019.

[45] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

[46] Michael S. Ryoo, A. J. Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In *Int. Conf. Learn. Represent.*, 2020.

[47] Jie Song, Limin Wang, Luc Van Gool, and Otmar Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[48] Kai Su, Dongdong Yu, Zhenqi Xu, Xin Geng, and Changhu Wang. Multi-person pose estimation with enhanced channelwise and spatial information. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[49] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[50] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[51] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[52] Manchen Wang, Joseph Tighe, and Davide Modolo. Combining detection and tracking for human pose estimation in videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[54] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[55] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[56] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Eur. Conf. Comput. Vis.*, 2018.

[57] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. Pose flow: Efficient online pose tracking. In *Brit. Mach. Vis. Conf.*, 2018.

[58] Lumin Xu, Ruihan Xu, and Sheng Jin. Hieve acm mm grand challenge 2020: Pose tracking in crowded scenes. In *ACM Int. Conf. Multimedia*, 2020.

[59] Sen Yang, Wankou Yang, and Zhen Cui. Pose neural fabrics search. *arXiv preprint arXiv:1909.07068*, 2019.

[60] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Int. Conf. Comput. Vis.*, 2019.

[61] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *Eur. Conf. Comput. Vis.*, 2020.

[62] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Int. Conf. Comput. Vis.*, 2013.

[63] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[64] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *Int. Conf. Learn. Represent.*, 2017.

[65] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

Appendix

A. Implementation Details

A.1. Spatial Search Space

We give implementation details of the spatial search space of ViPNAS in this section.

Elastic Depth. Elastic depth allows dynamic numbers of blocks in each stage. For example, the maximum number of the blacks in stage S is 4 as shown in Figure A1. When the depth D ($D \leq 4$) is selected, the first D blocks are activated and the rest $(4 - D)$ blocks are skipped. Note that the minimum depth of any stage should be no less than 1 ($D \geq 1$), as the first block may change the spatial resolution of the feature maps.

Elastic Width. Elastic width allows dynamic numbers of output channels in each block. For a convolutional layer, the shape of the filter is $O \times I \times K \times K$ given the input channels I, output channels O, and kernel size $K \times K$. When the output channel W ($W \leq O$) is selected, the filter is tailored to the shape of $W \times I \times K \times K$ as shown in Figure A2. We keep the first W out of O in the dimension of output channels.

Elastic Kernel Size. Elastic kernel size allows dynamic kernel sizes of convolutional layers in each block. The weights of the kernels are shared. As shown in Figure A4, we directly extract a $K \times K$ kernel filter from the centering of the super-network kernel filter, when the kernel size K is selected. This enables the weight sharing for kernels of different sub-networks, which has been shown simple but effective in our experiments. To avoid imbalance and biases of kernel extraction, we set the stride of the kernel size choice as 2, keeping all the selected kernels center-aligned.

Elastic Group Number. Elastic group number allows dynamic group numbers of convolutional layers in each block. A convolutional layer has a filter with the shape $O \times I \times K \times K$ given the input channels I, output channels O and kernel size $K \times K$. For example, when the group number is 2 (as shown in Figure A3), two filters with shape $\frac{O}{2} \times \frac{I}{2} \times K \times K$ are applied. In the figure, we concatenate the two groups of filters in the dimension of output channels for better illustration. We tailor the original filter to the shape of $O \times \frac{I}{2} \times K \times K$ and keep the first half in the dimension of input channels.

Elastic Attention Module. Elastic attention module allows the network to choose whether or not to use the attention module in each block. As shown in Figure A5, the attention module is used if attention module is selected. We skip the attention module and identity mapping is applied if attention module is not selected. The attention module will keep both the spatial resolution and the feature channels the same before and after.
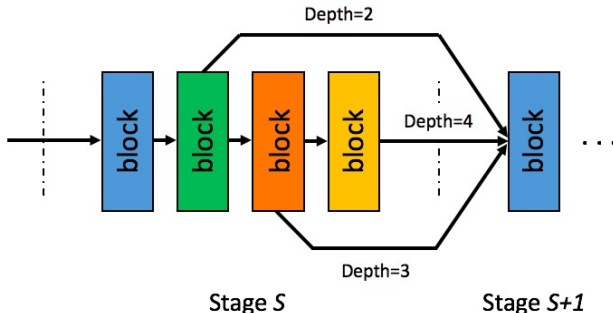

Figure A1. **Elastic Depth.** The first D blocks are activated if the depth D is selected in stage S.

A.2. Super-Network Design

In this section, we introduce the structure of our super-network as well as the concrete search space designs for each super-network. As the search space increases with the exponential explosion, directly searching for block-level network architecture is hard. In our experiments, we explicitly enforce the same width, kernel size, group number and attention module for all the blocks in the same stage and search for stage-wise optimum.

MobileNet-V3 [14]. Our MobileNet-V3 based super-network consists of one convolutional layer, six stages, and three deconvolutional layers (followed by one 1×1 convolutional layer for output). Each stage contains a stack with mobile blocks [14], which consists of one 1×1 expansion convolution, a middle convolution and one 1×1 projection convolution. We search for the kernel size and the group number of the middle convolution in mobile blocks. The expansion convolution expands the input features to a higher-dimensional feature space. We search the expansion ratio, which is similar to the elastic width. The detailed search space is summarized in Table A1.

ResNet-50 [13]. Following SBL [56], our ResNet-50 based super-network consists of one convolutional layer, four stages and three deconvolutional layers. Each block in stages is Bottleneck [13], which contains one 1×1 convolution followed by a middle convolution and another 1×1 convolution. Similar to our MobileNet-V3 based super-network design, we search for the kernel size and the group number of the middle convolution in the Bottleneck. We also search for whether to use a GC attention module [4] in each block. Table A2 specifies the search space of our ResNet-50 based super-network.

HRNet-W32[49]. We conduct experiments based on HRNet-W32 to further demonstrate the effectiveness of our proposed ViPNAS. Our HRNet-W32 based super-network consists of two convolutional layers followed by several Bottleneck blocks, three multi-resolution stages, and one
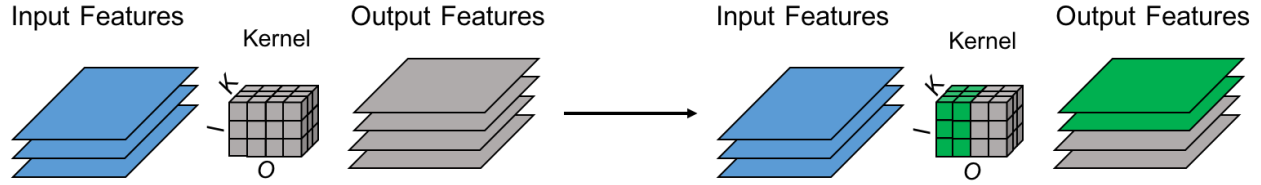
Figure A2. **Elastic Width.** Given the input channels I and kernel size $K \times K$, the first W output channels out of O is kept if the width W is selected. The filter is tailored from the shape $O \times I \times K \times K$ to $W \times I \times K \times K$.
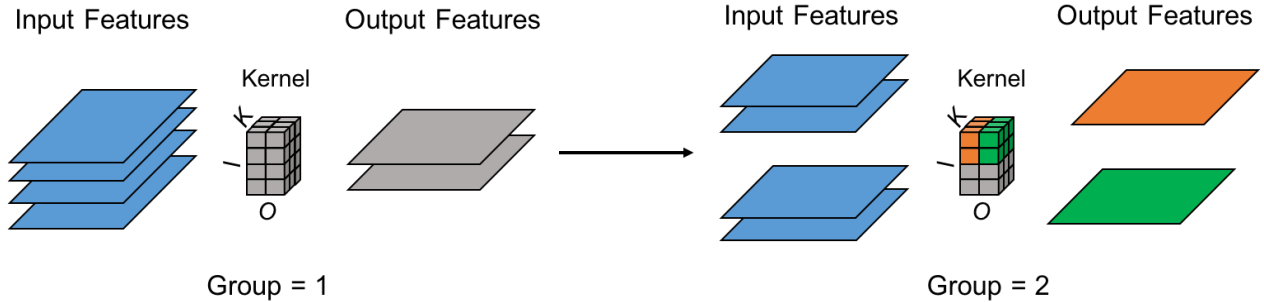

Group = 1   Group = 2

Figure A3. **Elastic Group Number.** An example of Group=2 is illustrated in the figure. Given the input channels I, output channels O, and kernel size $K \times K$, the filter is tailored from the shape $O \times I \times K \times K$ to $O \times \frac{I}{2} \times K \times K$. Two groups of filters with shape $\frac{O}{2} \times \frac{I}{2} \times K \times K$ are applied and are concatenated in the dimension of output channels.

Kernel 5x5   Kernel 3x3

Figure A4. **Elastic Kernel Size.** The centering $K \times K$ kernel is reserved if the kernel size K is selected.

Figure A5. **Elastic Attention Module.** The attention module is applied if attention is selected, and is skipped if not.

lution, and each branch includes several BasicBlock [13]. Both the convolutions in BasicBlock apply the same width, kernel size, and group number. We search the configurations of each stage and each branch for the best performance. Table A3 displays the detailed search space of our HRNet-W32 based super-network.

Our search space is discrete. Take ResNet-50 backbone as an example, we set the search step to be 1 for depth, 16 for width, 2 for kernel size and 16 for group.

B. Qualitative Results

Figure A6 shows the qualitative results of our T-ViPNAS-Res50 on four adjacent frames. S-ViPNet localizes human poses on the first frame (key frame), and three different T-ViPNets propagate poses on the following frames (non-key frame). Our lightweight models keep the temporal consistency and are robust to occlusion, motion blur and unusual illumination. ViPNAS achieves state-of-the-art accuracy with CPU real-time performance.

1×1 convolutional head for output. Each multi-resolution stage contains parallel branches with different spatial reso-

Table A1. **MobileNet-V3 [14]** based search space. [min, max] indicates the range of each search space. Expansion ratio indicates the feature channel expansion rate in the middle of mobile blocks, and resolution indicates the ratio between the shapes of current features and those of input images. Kernel size and group number of the middle convolution in mobile blocks are searched.
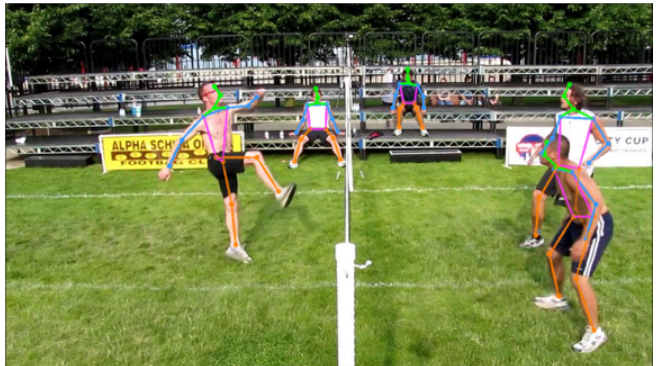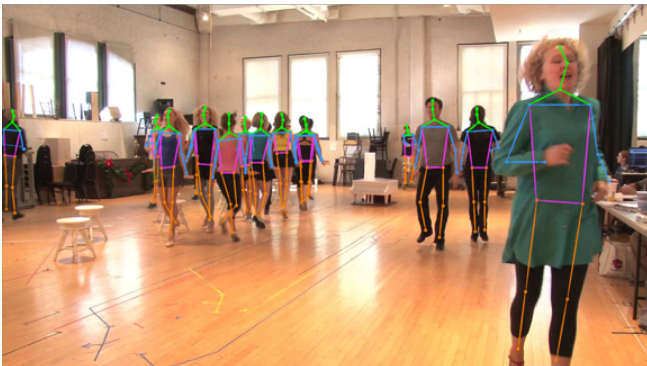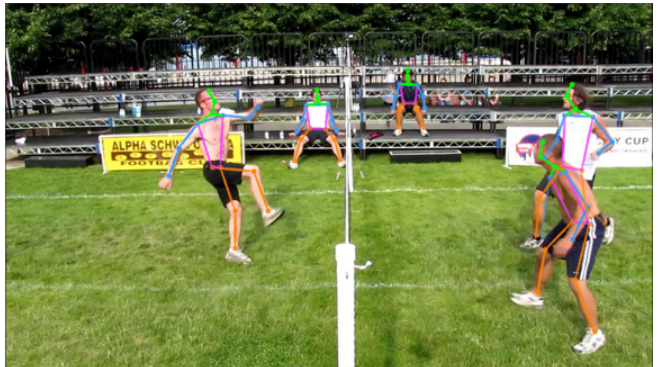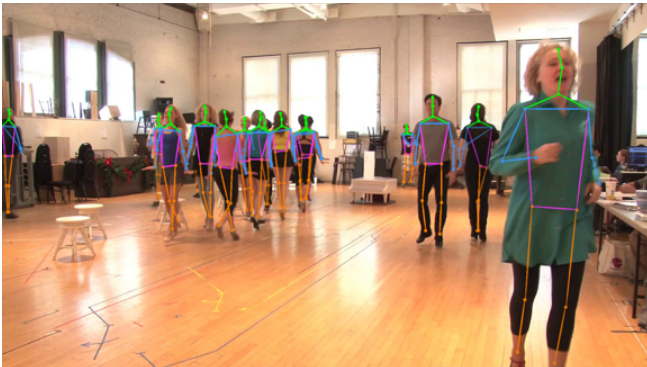
Stage	Operator	Depth	Width	Kernel Size	Group	Attention (SE [16])	Expansion Ratio	Resolution
	Conv	-	[16, 16]	[3, 3]	[1, 1]	-	-	1/2
1	Mobile Block	[1, 1]	[16, 16]	[3, 3]	[2, 16]	[0, 1]	[1, 1]	1/2
2	Mobile Block	[2, 4]	[24, 24]	[3, 7]	[9, 144]	[0, 1]	[3, 6]	1/4
3	Mobile Block	[2, 4]	[40, 40]	[3, 7]	[15, 240]	[0, 1]	[3, 6]	1/8
4	Mobile Block	[2, 4]	[80, 80]	[3, 7]	[30, 480]	[0, 1]	[3, 6]	1/16
5	Mobile Block	[2, 4]	[112, 112]	[3, 7]	[42, 672]	[0, 1]	[3, 6]	1/16
6	Mobile Block	[2, 4]	[160, 160]	[3, 7]	[60, 960]	[0, 1]	[3, 6]	1/32
	Deconv	-	[256, 256]	[4, 4]	[32, 256]	-	-	1/4

Table A2. **ResNet-50 [13]** based search space. [min, max] indicates the range of each search space, and expansion ratio indicates the feature channel expansion rate in the middle of Bottleneck. The first convolution and max pooling with stride 2 down-sample the spatial resolution to 1/4 of the input image. Kernel size and group number of the middle convolution in Bottleneck are searched.

Stage	Operator	Depth	Width	Kernel Size	Group	Attention (GC [4])	Expansion Ratio	Resolution
	Conv+Pool	-	[32, 64]	[7, 7]	[1, 1]	-	-	1/4
1	Bottleneck	[3, 4]	[64, 80]	[3, 5]	[16, 64]	[0, 1]	[1, 1]	1/8
2	Bottleneck	[4, 6]	[128, 160]	[3, 5]	[16, 64]	[0, 1]	[1, 1]	1/8
3	Bottleneck	[6, 8]	[256, 320]	[3, 5]	[16, 64]	[0, 1]	[1, 1]	1/16
4	Bottleneck	[3, 4]	[512, 640]	[3, 5]	[16, 64]	[0, 1]	[1, 1]	1/32
	Deconv	-	[64, 256]	[4, 4]	[16, 64]	-	-	1/4

Table A3. **HRNet-W32 [49]** based search space. HRNet includes parallel branches with different resolution in stages, which indicates the ratio between the spatial shape of current features and input images. We search depth of each stage, and search width and attention of each branch. Kernel size and group number of the middle convolution in Bottleneck and both the convolutions in BasicBlock are searched.

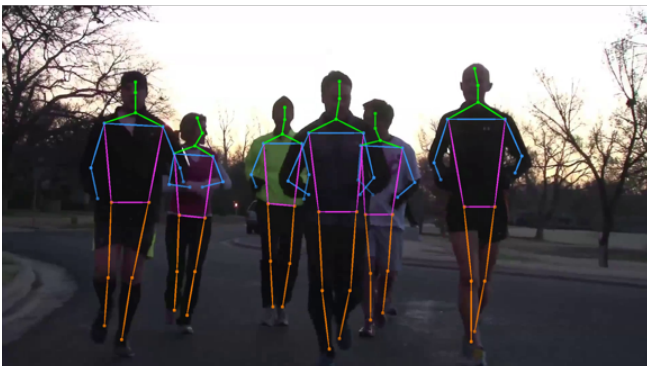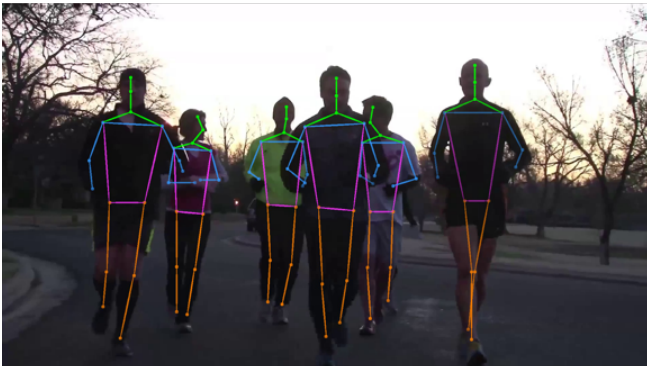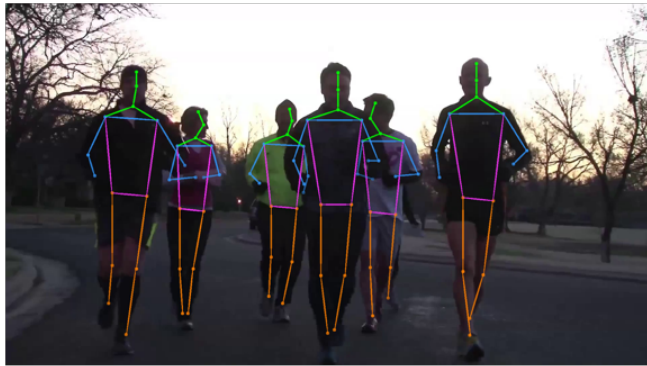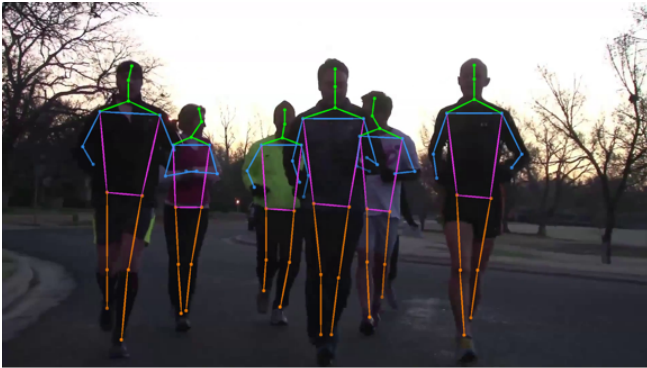| Stage	Depth	Branch	Operator	Width	Kernel Size	Group	Attention (SE [16])	Resolution
	-		Conv	[16, 64]	[3, 3]	[1, 1]	-	1/4
1	[2, 4]	1	Bottleneck	[16, 64]	[3, 3]	[1, 16]	[0, 1]	1/4
2	[4, 4]	1	BasicBlock	[8, 32]	[3, 3]	[1, 32]	[0, 1]	1/4
		2	BasicBlock	[16, 64]	[3, 3]	[1, 64]	[0, 1]	1/8
3	[8, 16]	1	BasicBlock	[8, 32]	[3, 3]	[1, 32]	[0, 1]	1/4
		2	BasicBlock	[16, 64]	[3, 3]	[1, 64]	[0, 1]	1/8
		3	BasicBlock	[32, 128]	[3, 3]	[1, 128]	[0, 1]	1/16
4	[8, 12]	1	BasicBlock	[8, 32]	[3, 3]	[1, 32]	[0, 1]	1/4
		2	BasicBlock	[16, 64]	[3, 3]	[1, 64]	[0, 1]	1/8
		3	BasicBlock	[32, 128]	[3, 3]	[1, 128]	[0, 1]	1/16
		4	BasicBlock	[64, 256]	[3, 3]	[1, 256]	[0, 1]	1/32

Figure A6. Qualitative results of T-ViPNAS-Res50 on four adjacent frames. S-ViPNet localizes human poses on the first frame, and three different T-ViPNets propagate poses on the following frames. Our proposed ViPNAS is robust to occlusion, motion blur and unusual illumination, and achieves state-of-art accuracy with CPU real-time performance.