CS172 Computer Vision I: Video Human Pose Estimation based on DEKR

Bi Chunhao, Yang Hongdi, Wu Qinman, Li Jiaxuan
Shanghaitech University
{bichh, yanghd, wuqm, lijx2}@shanghaitech.edu.cn

Abstract

This is the course project of CS172 Computer Vision I. Here we took the pose estimation topic, and performed some modifications on one of the cutting-edge algorithm. We used a dataset taken in Shanghaitech MARS lab to train our model, and performed some performance test. Because of the limited time, the model is not yet perfect, but the modification done in network structure and training adjustment can be used in later improvement. Our trained model is already capable for getting a quite good estimation.

1. Introduction

Human pose estimation has been a popular and challenging topic in computer vision. Given a 2D image, the goal is to estimate the keypoint positions for each human instance. This project mainly focuses on single-person video pose estimation, which is a representative case and can be extended to other more complicated scenarios such as multi-person video pose estimation. Current works exploit the information of a single picture, while in practical application, in many cases, it is necessary to carry out human pose estimation for multiple frames simultaneously. When multiple frames are involved, there still exists problems such as motion blur and occlusion. We aim at making full use of multi-frame information to achieve better performance in single-person video pose estimation, and our work is also applicable, theoretically, to multi-person video pose-estimation.

In this project, some supplements are made based on a bottom-up method called DEKR(Disentangled Keypoint Regression)[8], with a purpose to improve the pose estimation performance in videos. We propose a network structure based on DEKR, with a GRU unit added to attach temporal dependencies to the existing network.

DEKR(Disentangled Keypoint Regression)[8], which is a state-of-the-art strategy that has been introduced in CVPR 2021 by Geng et al. The DEKR method adopts adaptive convolutions, through pixel-wise spatial transformer to activate the pixels lying in the keypoint regions. Compared with the conventional convolutions, it can make the activation range of pixels no longer limited to the neighbors of central pixel, concentrate on the surrounding regions keypoints instead. Then the features and representations of these activated regions are learned. Furthermore, a multi-branch structure is designed to do the regression to decouple the representations of every keypoint. Each branch is associated with an adaptive convolution concentrated on a single keypoint, to regress the position of corresponding keypoint based on the features learned.

However, in video pose estimation, instead of predicting every frame of the video individually, RNN(Recurrent Neural Network) is used to strengthen the relation between continuous video frames. GRU (Gated Recurrent Unit) as an alternative form of gated RNN, is a useful method for handling sequential data. A single video can be parsed as a sequence of images which have high similarities in adjacent frames. Therefore, RNN is suitable for doing the prediction based on former inputs. GRU provides a way to decide when to forget the information in previous time. It has fewer parameters to update and thus is more efficient compared to LSTM(Long-Short Term Memory), which is another method in RNN.

2. Related Work

We investigated the related work based on the multi person pose estimation problem, because this is one of the most cutting-edge fields at present. However, due to insufficient time for experiments, our model was finally trained on a single person dataset.

There are two main methods for multi-person human pose estimation: top-down and bottom-up. The top-down method detects the person in the image in the first step, and then performs single-person pose estimation for each human instance. It is costly since object detection is involved as a preprocessing step. The bottom-up implementation, however, finds all the keypoints first, and then grouping them into different person. In most cases this method yields a better efficiency, which makes it possible to do human pose estimation in real-time.

2.1. Top-down methods

Top-down methods transform the problem of multi-person pose estimation into the problem of single person pose estimation by detecting a single person within a person bounding box first, followed by estimation of the body joint keypoints within that box. The bounding boxes are generated by object detectors such as Faster R-CNN[26], Mask R-CNN[10]. Top-down works include: HRNet[29, 32], PoseNet[24], RMPE[7], convolutional pose machine[34], Hourglass[21], Mask R-CNN[10], CPN[3], simple baseline[35], Graph-PCNN[31],RSN[1], and so on. These methods exploit the advances in person detection, achieving satisfactory performance, but the runtime suffers. For each person bounding box, a single-person pose estimator is run, and the more people there are, the greater the cost be.

2.2. Bottom-up methods

Many bottom-up methods firstly detect all keypoints in the picture, and then group the keypoints belonging to the same person. Various grouping techniques are developed. The pioneering work, DeepCut [25], DeeperCut[12], and L-JPA[13] formulate the keypoint association problem as an integer linear program. Openpose[2] proposes part affinity fields, which is extended in PifPaf[16]. Associative embedding[20] is an end-to-end joint detection and grouping method, and is also used in HigherHRNet[4]. PersonLab[23] proposes greedy decoding with Hough voting, and HGG[15] proposes graph clustering.

Another kind of bottom-up methods directly regresses the keypoint positions belonging to the same person. The pioneering works of multi-person pose estimation problem such as Deeppose[30] regress coordinates directly, while most of the current works take Heatmap as the output of the network. Several recent works[36, 22, 33], such as CenterNet[6] densely regress a set of pose candidates, followed by a post-processing scheme, matching the regressed keypoint positions to the closest keypoints detected from the keypoint heatmaps. Disentangled keypoint regression[8] directly uses regression results because this work ensures the position prediction for one keypoint spatially accurate by learning disentangled representations.

These works exploit the information of a single picture, but in practical application, in many cases, it is necessary to carry out real-time human pose estimation for multi-frame information. There still exists problems of low frame rate and can not deal with motion blur and occlusion well. We hope to make full use of multi-frame information to deal with motion blur and occlusion on the basis of studies above, in order to achieve higher accuracy in the problem of multi-person pose estimation in videos.

2.3. Pose tracking in videos

Some top-down pose tracking works go further into pose tracking[11, 14, 28, 9, 35], and the main method is to assign a single label to the same person detected in multiple frames by adding a RNN[19, 27]. The PoseTrack[14] dataset provides labels for different joints of the human body to evaluate the pose tracking task by Multiple Object Tracking (MOT) metrics[18]. We believe that connecting each joint temporally is also helpful to increase the accuracy of pose estimation, especially when there exists motion blur or occlusion, resulting in failures of keypoint detection. So we aim to apply methods of multi-object tracking task to human pose estimation.

The most common framework used in MOT is tracking by-detection strategy which links detections across frames by data association algorithms. For data association, either in traditional methods or deep learning methods, most existing works realize data association by motion model, appearance model or composite model. GAKP[17] integrate the auto-tuning Kalman method for prediction step and GRU for the association step. The link probability between predictions and detections is predicted with non-linear combination of motion and appearance features.

Our approach aims to make use of the methods of multi-object tracking to deal with the problem of motion blur and occlusion in pose estimation which are quite common in practical application. We transform the multi-frame pose estimation problem into the object tracking problem for each joint of multiple person, and propose a novel network based on Disentangled keypoint regression[8], with a GRU[5] unit added to model temporal dependencies.

3. Methodology

For human pose estimation task, the aim is to predict the K keypoints of a person via the network. In COCO dataset standard, each human pose consists of 17 keypoints. Due to training dataset limitation, instead of focusing on multi-person, we try to give a method for better single-person pose estimation in videos.

3.1. HRnet

Same as DEKR[8], we use HRnet[29] as backbone for feature extraction. The advantage of HRnet is that it connects high-to-low subnetworks in parallel, so it maintains high-resolution representations through the whole process for spatially precise heatmap estimation.

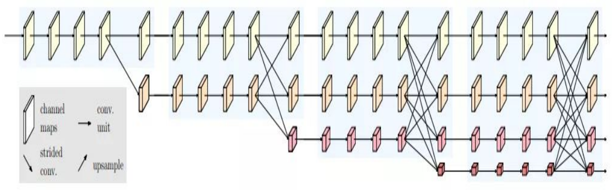

Figure 1. Basic structure for HRnet

The key part of HRnet is its fusion layer, the feature maps in every branch will be fused to another branch in fusion layer. To explain the fusion process simply, there are two types of fusion.

- Fusion from lower resolution to higher resolution, we use a 1X1 Conv block and upsampling to match the size and channels.
- Fusion from higher resolution to lower resolution, we use a 3X3 Conv block to downsample. For last conv block, we change the channels to match that with lower resolution.

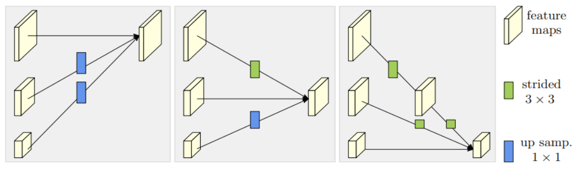

Figure 2. Fusion layer for HRnet

In this project, HRnet-W32 is used, which has 4 branches with channels 32, 64, 128, 256.

3.2. DEKR

To predict keypoints position more accurately, DEKR uses an approach named Disentangled Keypoint Regression. It uses a multi-branch structure for separate regression: each branch learns a representation with dedicated adaptive convolutions and regresses one keypoint.

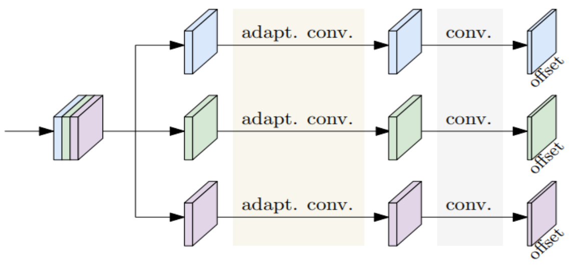

Figure 3. Basic layer for DEKR

As we use Coco dataset standard in this project, so there are 17 branches for regressing the 17 keypoints. We represent every branch with a function

$$O = F(X)$$

where X is the feature computed from HRnet, $F()$ is the keypoint position regression head predicting the offset maps O. So in total we have 17 functions.

$$O_1 = F_1(X)$$
$$O_2 = F_2(X)$$
$$\vdots$$
$$O_{17} = F_{17}(X)$$

The 17 regression functions have the same structure, while the parameters for each regression function are learned independently.

In feature extraction part, DEKR uses adaptive convolution. It is similar to deformable convolution, which changes the receptive field of each pixel. The 3×3 kernel for each pixel uses an affine transform to acquire the bias of the kernel.

$$y(q) = \sum_{i=1}^{9} W_i x(g_{si}^q + q)$$
$$\{g_{s1}, g_{s2} \dots g_{s9}\} = A^q G_t + [t\ t\ \dots\ t]$$

where

$$G_t = [(-1,-1)^T, (0,-1)^T, (1,-1)^T \dots (1,1)^T]$$

Here q denotes the 2D pixel position, A is the affine matrix estimated in the essay and t is a translation vector. It then uses the adaptive kernel to calculate the value. This makes it possible for each pixel to concentrate not in its neighborhood, but more around the keypoints.

3.3. DEKR with GRU

The origin DEKR network is constructed specifically for image pose estimation. The raw video test in DEKR network is carried out by separating a whole video into frames and each frame are calculated independently. However, if the temporal information of frames are taken into consideration, maybe the network will result in better performance.

Therefore, to acquire temporal relation, we attempted a simple method. A GRU module is added at the end of the origin network, which takes the former input into consideration. The network structure is modified as well, from loading single images to image batches. The images in the same batch are carried through the network together to acquire the origin output. Then they go through the GRU in order, and the module stores the information between the sequential images. The module will store useful temporal information automatically.

Figure 4. shows the structure of GRU. Hidden state h^t represent the information stored from former input, x^t in this case represent the origin estimation, and y^t represent the new result.

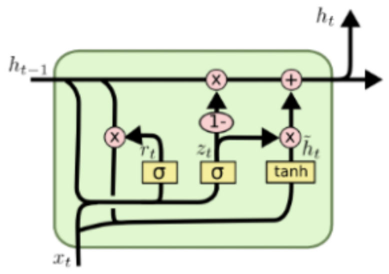

Figure 4. GRU

To be more detailed, GRU contains two gates - reset gate and update gate. The reset gate decides how much information to be stored in the memory $\widetilde{h}_t$, and the update gate determine the information weight for the next input in r_t and z_t. The output and h_t depend on those parameters from last input with $t-1$ index.

3.4. Loss Function

For the loss function, we use a two-part loss function which is the sum of heatmap loss and the regression loss.

$$l = l_h + \lambda l_p$$

where l_h denotes the heatmap loss and l_p denotes the regression loss. λ is a trade-off weight which set to 0.03.

Regression loss

$$l_p = \sum_{i \in C} \frac{1}{Z_i} smooth_{L_1}(o_i - o_i^*)$$

where the normalized smooth loss is used to form the pixel-wise keypoint regression loss. $Z_i = \sqrt{H_i^2 + W_i^2}$ is the size of the corresponding bounding box for the detected person. C is the set of the positions that have groundtruth poses. o_i is the 2x17-dimensional estimated offset vector for the position i, while o_i^* is the groundtruth offset vector for the position i.

Keypoint and center heatmap estimation loss

To score and rank regressed poses, 17 keypoint heatmaps is estimated using a heatmap estimation branch. Each heatmap corresponds to a keypoint type and the center heatmap indicates the confidence that each pixel is to be the center of some person.

$$(H, C) = H(X)$$

Then we calculate the heatmap loss by weighted distances between the predicted heat values and the groundtruth heat values.

$$l_h = ||M^h \odot (H - H^*)||_2^2 + ||M^C \odot (C - C^*)||_2^2$$

where $||\cdot||_2$ is the entry-wise 2-norm. $\odot$ is the Hadamard product. M^h has 17 masks, and the size is H x W x 17. The kth mask, M_k^h is formed so that the mask weight of the positions not lying in the kth keypoint region is 0.1, and others are 1. And for center heatmap mask M^c, it is the same. H^* and C^* are the groundtruth of target keypoint and center heatmaps.

4. Results and Experiments

4.1. Training

DEKR is implemented with COCO and Crowdpose dataset format, which use COCO APIs. However, those are just images, which are not suitable for this project. Therefore, we used a dataset recorded in Shanghaitech MARS lab to train our model. The videos and ground truths are converted into COCO format and then are used into model training. The model also need modifications to load image batches. After all the alternatives done, the model can be trained successfully.

When training, random rotation ([-30°, 30°]), random scale ([0.75, 1.5]) and random translation ([-40, 40]) are performed, which is the same as the data augmentation in original DEKR.

As HRNet-W32 is used in this project, the input image is cropped into 512 x 512 during training. Adam optimizer is used in the training process, and the learning rate is set to the same as original DEKR.

Due to the limited time and the requirement of abundant data for the model, we only trained the model with 30 epochs on a subset of whole data.

4.2. Testing

As COCO dataset is too large and not suitable for video testing, we still use Dataset recorded in Shanghaitech MARS lab for testing.

During testing, we also resize the images to 512X512. Multiscale tests are not going to be performed.

Evaluation Metrics

Performing testing on single-person pose estimation on coco dataset standard, we use OKS (object keypoint similarity) score on single person.

Also, we use the NMS score introduced in DEKR to show the average of the heat values at the regressed K keypoints. Though the NMS score does not directly tell the accuracy, it can help to estimate the local prediction accuracy without ground truth.

4.3. Results

For single video testing, we input a video and images with drawn skeletons will be output, along with a complete video containing the skeletons.

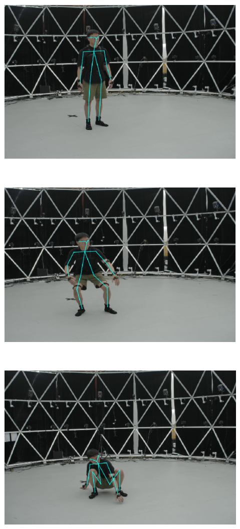

Figure 5. Testing our model on person face front

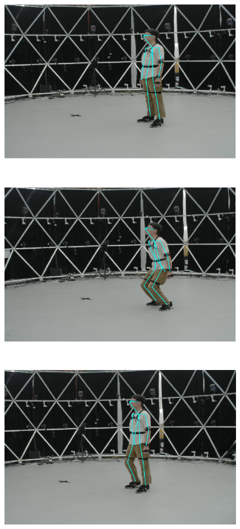

Figure 6. Testing our model on person face aside

Figure 5.6.are snapshots of the output videos. When testing with person who only show one side of body most of the time, the result would be worse compared that to face-fronted person. The reason could be that one arm of the person is almost totally occluded, thus making it difficult to predict the person's pose.

To better estimate the difference, we test our model on two different test sets with person front-faced and person side-faced. We get following metrics.

Table 1. Testing result on two different test sets

test set	local NMS score	OKS score	AP
person front-faced	77.15	65.97	61.2
person side-faced	62.62	55.93	53.4

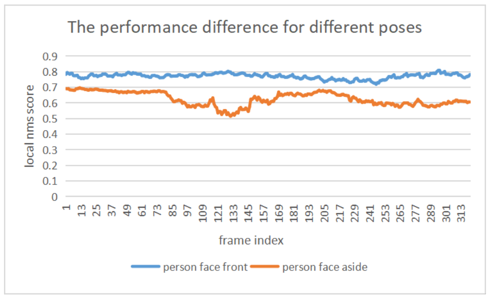

Figure 7. change of local NMS score with frame index

It could be easily to see that the performance on front-faced person is higher than that on side-faced person. And the performance on front-faced person is also more stable.

4.4. Performance Evaluation

However, the performance actually does not improve much compared to the original DEKR network, and in some scenes even worse. Here we do a performance evaluation on the possible reasons that cause this result.

- The training epoch is not enough. Due to time limitation, we only trained for 30 epochs on a subset of the whole data, which may not be enough for the GRU to optimize.
- The training data in training dataset is not enough. In the training dataset, there are only single-person data. The scene is also duplicated, which may make the model less robust.

4.5. Discussion

During the experiment, we noticed that the result of front-faced person pose estimation is much better than that of side-faced human pose estimation, which suggests that future work may try to carry out specific research and design for lateral human body in order to achieve better results.

5. Conclusion

In this project, we present a video human pose estimation method based on DEKR associated with a GRU module. Although there is only limited time for training, our model can generally yield a quite good pose estimation result on our testing set, with a few frames having worse performance than original DEKR model. The temporal information of video is taken into consideration in our GRU structure, which can cooperate with DEKR to make better evaluation on video materials. We believe that our idea above can produce a satisfying model for video human pose estimation, and also can be inspiring for future relative research.

References

[1] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xiangyu Zhang, Xinyu Zhou, Erjin Zhou, and Jian Sun. Learning delicate local representations for multi-person pose estimation, 2020.

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields, 2017.

[3] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation, 2018.

[4] 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, 2020.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[6] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection, 2019.

[7] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation, 2018.

[8] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression, 2021.

[9] Rohit Girdhar, Georgia Gkioxari, Lorenzo Torresani, Manohar Paluri, and Du Tran. Detect-and-track: Efficient pose estimation in videos, 2018.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[11] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov, Bjoern Andres, and Bernt Schiele. Arttrack: Articulated multi-person tracking in the wild, 2017.

[12] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model, 2016.

[13] Umar Iqbal and Juergen Gall. Multi-person pose estimation with local joint-to-person associations, 2016.

[14] Umar Iqbal, Anton Milan, and Juergen Gall. Posetrack: Joint multi-person pose estimation and tracking, 2017.

[15] Sheng Jin, Wentao Liu, Enze Xie, Wenhai Wang, Chen Qian, Wanli Ouyang, and Ping Luo. Differentiable hierarchical graph grouping for multi-person pose estimation, 2020.

[16] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation, 2019.

[17] Zhen Li, Sunzeng Cai, Xiaoyi Wang, Zhe Liu, and Nian Xue. Gakp: Gru association and kalman prediction for multiple object tracking, 2020.

[18] Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking, 2016.

[19] Anton Milan, Seyed Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks, 2016.

[20] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping, 2017.

[21] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation, 2016.

[22] Xuecheng Nie, Jianfeng Zhang, Shuicheng Yan, and Jiashi Feng. Single-stage multi-person pose machines, 2019.

[23] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model, 2018.

[24] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild, 2017.

[25] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation, 2016.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[27] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies, 2017.

[28] Jie Song, Limin Wang, Luc Van Gool, and Otmar Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos, 2017.

[29] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation, 2019.

[30] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014.

[31] Jian Wang, Xiang Long, Yuan Gao, Errui Ding, and Shilei Wen. Graph-pcnn: Two stage human pose estimation with graph pose refinement, 2020.

[32] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition, 2020.

[33] Fangyun Wei, Xiao Sun, Hongyang Li, Jingdong Wang, and Stephen Lin. Point-set anchors for object detection, instance segmentation and pose estimation, 2020.

[34] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines, 2016.

[35] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking, 2018.

[36] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points, 2019.