Adaptive Graphical Model Network for 2D Handpose Estimation

Deying Kong[1]
deyingk@uci.edu
Yifei Chen[2]
chenyifei.star@gmail.com
Haoyu Ma[3]
howiema@seu.edu.cn
Xiangyi Yan[4]
11510706@mail.sustech.edu.cn
Xiaohui Xie[1]
xhx@ics.uci.edu

[1] University of California, Irvine
[2] Tencent
[3] Southeast University
[4] Southern University of Science and Technology

Abstract

In this paper, we propose a new architecture called Adaptive Graphical Model Network (AGMN) to tackle the task of 2D hand pose estimation from a monocular RGB image. The AGMN consists of two branches of deep convolutional neural networks for calculating unary and pairwise potential functions, followed by a graphical model inference module for integrating unary and pairwise potentials. Unlike existing architectures proposed to combine DCNNs with graphical models, our AGMN is novel in that the parameters of its graphical model are conditioned on and fully adaptive to individual input images. Experiments show that our approach outperforms the state-of-the-art method used in 2D hand keypoints estimation by a notable margin on two public datasets.

1 Introduction

Understanding human hand pose is a critical task for many real world AI applications, such as human-computer interaction, augmented reality and virtual reality. However, hand pose estimation remains very challenging because the hand is highly articulated and dexterous, and hand pose estimation suffers severely from self-occlusion. An intuitive approach is to resort to multi-view RGB cameras [13, 19], which unfortunately requires expensive hardware and strict environment configurations. For practical daily applications, many researchers have also explored the problem under monocular RGB [2, 16, 28] or RGB-Depth [1, 23, 27] scenarios. Solving 3D pose estimation problem [2, 28] often relies on 2D hand pose estimation, making 2D hand pose estimation itself an important task. In this paper, we focus on the task of 2D hand pose estimation from a monocular RGB image.

The advent of Deep Convolutional Neural Networks (DCNNs) has enabled this field to make big progress in recent years. For example, the Convolutional Pose Machine (CPM) [25] is one of the most successful DCNNs that have been applied to 2D hand pose estimation

[19], although it was originally proposed for the task of human pose estimation. However, despite the fact that DCNNs like CPM have the power to learn good feature representations, they often fail to learn geometric constraints among joints, resulting in joint inconsistency in the final prediction as observed in human pose estimation tasks [14, 21]. For 2D hand pose estimation, the situation could be even worse, since there are more articulations and self-occlusion is severer.

To model the relationships among joints, several studies have also explored the possibility of combining DCNN and the Graphical Model (GM) in pose estimation tasks. Existing methods [21, 22, 26] impose a self-independent GM on top of the score maps regressed by DCNNs. The parameters of the GM are learned during end-to-end training, then these pairwise parameters are fixed and shared by all input images during inference.

In this paper, we propose the Adaptive Graphical Model Network (AGMN), which is a brand new framework for combining DCNNs and GM. By "adaptive", we mean that the parameters of the GM should be able to adapt to individual input images, instead of being fixed and shared by all input images or among a group of input images. The adaptivity of the GM is achieved by setting the pairwise parameters of the GM to be the output of a DCNN whose input is the image. Meanwhile, the unary potentials (score maps of each hand joint location) are regressed from another branch of DCNN. Then, final score maps are inferred by the GM using techniques like message passing. The whole AGMN architecture could be trained end-to-end.

We show the efficiency of our proposed framework on two public datasets: the CMU Panoptic Hand Dataset [19] and the Large-scale Multiview 3D Hand Pose Dataset[7]. Our approach outperforms the popularly used algorithm CPM by a noticeable margin on both datasets. Qualitative results show our model could alleviate geometric inconsistency among predicted hand keypoints significantly when severe occlusion exists.

The main contributions of this work are:

- We propose a novel framework integrating DCNNs and graphical model, in which the the graphical mode parameters are fully adaptive to individual input images, instead of being shared among the input images.
- By implementing the message passing algorithm as a sequence of 2D convolutions, the inference is performed efficiently and the AGMN could be trained end-to-end.
- Our AGMN could reduce the inconsistency and ambiguity of hand keypoints significantly in scenarios of severe occlusion, as shown by experiments on two real world hand pose datasets.

2 Related Work

2.1 Human pose estimation

Research on hand pose estimation has benefited from the progress in the study of human pose estimation. On one hand, DCNNs have been successfully applied to human pose estimation [3, 6, 9, 24] in recent years. The DCNN-based algorithms are typically equipped with well crafted deep architectures [8, 20] and/or multi-stage training technique[15, 25]. Since DCNNs have large receptive fields, they could learn salient and expressive feature representations. However, DCNNs could only capture structural constraints among body parts implicitly, resulting in limited performance in practice when severe occlusion and cluttering exist [14, 21]. Some approaches try to learn extra tasks (*e.g.*, offset fields [17], compound

heatmaps [14]) besides heatmaps of joint positions, with the purpose of providing more additional structural information. Nevertheless, these methods still could not fully exploit structural information.

On the other hand, there is also a trend to combine DCNN with graphical model for pose estimation [22, 23, 26], recently. It has been studied in several scenarios, i.e., human pose estimation in a vedio[21], multi-person pose estimation [10, 18], multi-person pose tracking [11, 12] and also in other fields, e.g., word prediction and image tagging[4]. However, in all of the above approaches, graphical model parameters are not conditioned on individual input images. Authors in [5] propose to select parameters of graphical models basing on different categories of the relationships between neighboring joints. Although the joint relationships are dependent on input images, all the graphical model parameters are still fixed and shared among different input images after training. The graphical model parameters are not fully adaptive to individual images. Also the model in [5] does not support end-to-end training.

2.2 Hand pose estimation

The 3D hand pose estimation is a challenging task due to strong articulation and heavy self-occlusion of hands. Some researcher try to solve the task efficiently with the help of multi-view RGB cameras [13, 19]. However, this kind of approaches are impractical for daily applications as they require expensive hardware and strict environment configurations. To circumvent this limitation, other studies have been focused on depth-based solutions [1, 23, 27] where RGB-D cameras are used. Due to the ubiquitousness of regular RGB cameras, researchers also have a great interest in solving hand pose estimation from monocular RGB images [2, 16, 28].

2D hand pose estimation plays an important role in the task of estimating 3D hand pose, since 3D estimation is often inferred from 2D estimation[2, 16, 28]. Current algorithms on 2D hand pose estimation often directly deploy DCNN-based human pose estimators. Among a variety of DCNN-based models, CPM is commonly used in 2D hand pose estimation[2, 16, 19, 28], yielding state-of-art performance. Thus, in this work, we choose CPM as the baseline for comparison with our proposed model.

3 Method

3.1 Basic Framework of Adaptive Graphical Model Network

As shown in the left image of Fig. 1 (a), due to the lack of explicit structural information, CPM fails when the hand is occluded severely, resulting in hand keypoints' spatial inconsistency. To alleviate this problem, we propose the novel Adaptive Graphical Model Network (AGMN), the efficiency of which could be seen from the right image of Fig. 1 (a). The model contains two DCNN branches, the *unary branch* and the *pairwise branch*, and a graphical model inference module, as depicted in Fig. 1 (b). The unary branch would output intermediate score maps of K hand keypoints. Existing DCNN-based pose estimators that regress score maps could be fitted into our AGMN as the unary branch easily. The pairwise branch produces parameters that characterize pairwise spatial constraints among hand keypoints. These parameters would be later used in the graphical model. It is the pairwise branch that makes our model distinguish from existing models [5, 21, 22, 26] which also try to combine graphical model with DCNNs. In our approach, the parameters of the graphical model are

not independent parameters. Instead, they are closely coupled with the input image via a DCNN and they are adaptive to different input images. In approaches from [6, 21, 22, 26], once the graphical model parameters are learned, they would be fixed and used for different input images in future prediction.

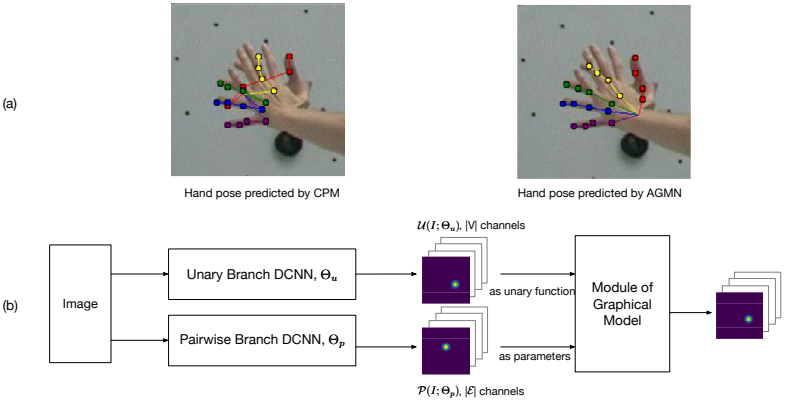


Figure 1: Basic flow diagram of adaptive graphical model network.

The hand pose estimation problem could be formulated by using a graph. Let $G = (V, \mathcal{E})$ denote a graph with a vetex set V and an edge set $\mathcal{E}$, where $V = \{v_1, v_2, \cdots, v_K\}$ corresponds to the set of K hand keypoints and $\mathcal{E} \subseteq V \times V$ is the set of edges between neighboring keypoints. Let the discrete variable $x_i \in \mathbb{R}^2$ denote the 2D position of the keypoint associated with v_i.

The joint probability distribution of a hand pose configuration is given by

$$p(X|I;\Theta) = \frac{1}{Z} \prod_{i=1}^{|V|} \phi_i(x_i|I;\Theta_u) \prod_{(i,j)\in\mathcal{E}} \varphi_{i,j}(x_i, x_j|I;\Theta_p), \tag{1}$$

where $X = \{x_1, x_2, \cdots, x_K\}$ represents positions of all the keypoints, I stands for the input image, $|V|$ is the cardinality of the set V and Z is the partition function. The whole set of AGMN's prameters Θ consists of two components, parameters for the unary branch and that for the pairwise branch, i.e., $\Theta = \{\Theta_u, \Theta_p\}$.

Unary Terms. The non-negative term $\phi_i(x_i|I;\Theta_u) \in \mathbb{R}$ is the local confidence of the appearance of the i-th keypoint at location x_i. Let $\mathcal{U}(I;\Theta_u) \in \mathbb{R}^{|V|\times h_u \times w_u}$ denote the output of the unary branch in Fig. 2, where w_u and h_u are the width and height of the output heatmap. We define

$$\phi_i(x_i|I;\Theta_u) = \max\left(0, \mathcal{U}^i_{x_i}(I;\Theta_u)\right), \tag{2}$$

where $\mathcal{U}^i_{x_i}(I;\Theta_u) \in \mathbb{R}$ is the value of the i-th channel of $\mathcal{U}(I;\Theta_u)$ evaluated at location x_i.

Pairwise Terms. The term $\varphi(x_i, x_j|I, \Theta_p) \in \mathbb{R}$ represents the pairwise potential function between the i-th and j-th keypoints, if (i, j) forms an edge in the graphical model. It encodes spatial constraints between two neighboring keypoints. The pairwise term is given by

$$\varphi_{i,j}(x_i, x_j|I;\Theta_p) = \mathcal{F}(x_i, x_j; \theta^{(i,j)}), \tag{3}$$

$$\theta^{(i,j)} = \max(0, \mathcal{P}^{(i,j)}(I;\Theta_p)), \tag{4}$$

where $\mathcal{P}(I;\Theta_p) \in \mathbb{R}^{|\mathcal{E}|\times h_p \times w_p}$ is the output of the pairwise branch in Fig. 2, $\mathcal{P}^{(i,j)}(I;\Theta_p) \in \mathbb{R}^{h_p \times w_p}$ is a channel of $\mathcal{P}(I;\Theta_p)$ corresponding to the pair of the i-th and j-th keypoints. Function $F(\cdot)$ is defined as $\mathcal{F}(x_i,x_j;\theta^{(i,j)}) = \theta^{(i,j)}_{x_i-x_j}$, which is an entry of the matrix $\theta^{(i,j)} \in \mathbb{R}^{h_p \times w_p}$, indexed by the relative position of the i-th keypoint with respect to the j-th keypoint.

One can also design $\theta^{(i,j)}$ as a set of parameters of a spring model , and then define $\mathcal{F}(\cdot)$ as a quadratic function as in [5, 21, 26] . In this work we follow the idea in [22] and design $\theta^{(i,j)}$ to be a 2D matrix, which has a much larger parameter space.

Inference. The final score maps generated by AGMN are the marginal distributions of $p(X|I;\Theta)$ given in Eq.(1). The marginals are defined as

$$p_i(x_i|I;\Theta) = \sum_{V\setminus x_i} p(X|I;\Theta), \tag{5}$$

which is computed in the module of graphical model. Finally, the predicted position of hand keypoint i is obtained by maximizing its marginal probability as

$$x_i^* = \underset{x_i}{\operatorname{argmax}}\, p_i(x_i|I;\Theta). \tag{6}$$

In summary, the complete parameters in the AGMN model is given by $\Theta = \{\Theta_u, \Theta_p\}$, consisting the parameters from the unary branch and pairwise branch.

3.2 Detailed Structure of AGMN

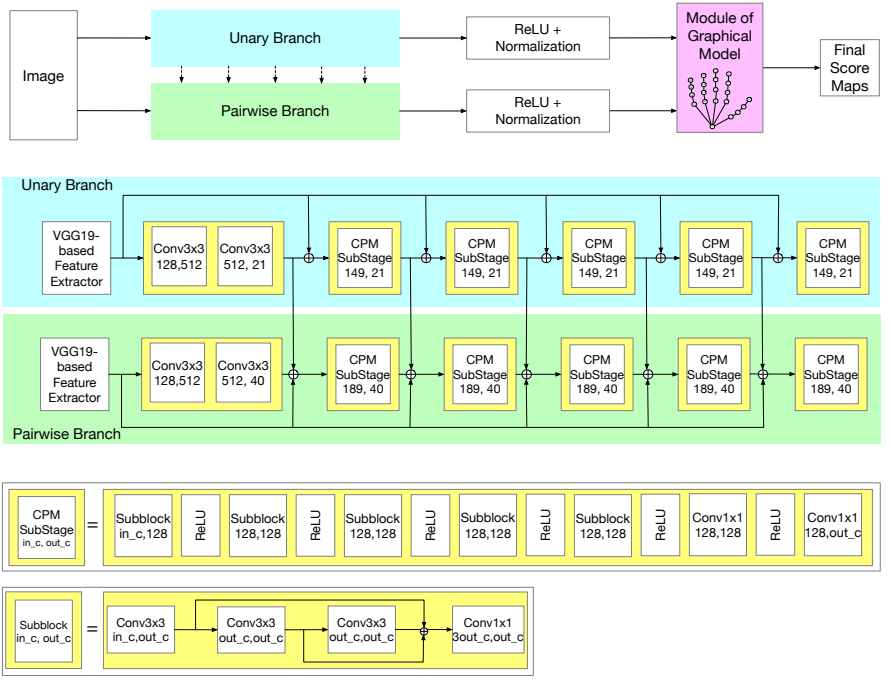


Figure 2: More detailed inllustration of adaptive graphical model network.

The detailed structure of AGMN is shown in Fig. 2.

Unary branch. The CPM architecture in [19] is used as the unary branch in our AMGN model, and it's also compared with as the baseline in our experiments. A pre-initialized VGG-19 network [20] up to conv4_4 and additional convolutions are used to produce the 128-channel features, then several prediction stages follows. The output of the unary branch is a 21-channel score map, each channel corresponding to one keypoint of the hand.

Pairwise branch. The pairwise branch follows the similar structure of the unary branch. The only difference is that the pairwise branch outputs a 40-channel kernel instead of a 21-channel score map. This 40-channel kernel would later be utilized in the module of graphical model. There are also some information flowing from the unary branch to the pairwise branch, as indicated by the arrows between the unary branch and pairwise branch in Fig. 2. We found that adding such information flows would benefit the performance.

Inference. *Message Passing.* Sum-product algorithm is widely used for efficient calculation of marginals in a graphical model. Vertices receive messages from and send messages to their neighbors. The sum-product algorithm updates the message sent from hand keypoint i to keypoint j as follows:

$$m_{ij}(x_j) = \sum_{x_i} \varphi_{i,j}(x_i, x_j)\phi_i(x_i) \prod_{k \in Nbd(i)\setminus j} m_{ki}(x_i). \tag{7}$$

Let M_{ij} denote the complete message sent from keypoint i to j, then $M_{ij} \in \mathbb{R}^{h_u \times w_u}$, since x_j could take values from a set of grid points which has the size of $h_u \times w_u$. After several iterations or convergence, the marginal probabilities are approximated by

$$p_i(x_i) \approx \frac{1}{Z'}\phi_i(x_i) \prod_{k \in Nbd(i)} m_{ki}(x_i), \tag{8}$$

where Z' is just a normalization term.

Tree Structured Graphical Model. In our implemented AGMN, a tree-structured graphical model is used. One advantage of tree-structured model is that exact marginal probability in Eq.(8) could be obtained by belief propagation. The tree-structured hand model is shown in Fig. 3. By passing messages from leaves to root and then from root to leaves, the exact marginal can be reached. The numbers along side each arrow in Fig. 3 indicates the schedule of massage updates. In total, we only need pass messages 40 times to obtain exact marginals.

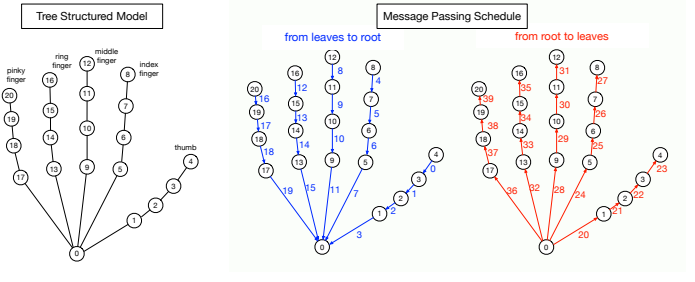


Figure 3: Tree structured model and message passing schedule.

Message updates as convolution operations. When implementing Eq. (7), one way to avoid the for loop in the summation is to use matrix product. However, if we write $\varphi_{i,j}(x_i, x_j)$

compactly into a matrix, the dimension of this matrix is huge. Since x_i and x_j could both take $h^u \times w^u$ different values, The matrix storing $\varphi_{i,j}(x_i, x_j)$ would have the size of $(h^u \times w^u)^2$. To save memories during the inference, we resort to convolution operations when performing message passing.

The message update formula in Eq. (7) could be rewritten as

$$m_{ij}(x_j) = \sum_{x_i} \varphi_{i,j}(x_i, x_j) h_i(x_i), \tag{9}$$

$$h_i(x_i) = \phi_i(x_i) \prod_{k \in Nbd(i) \setminus j} m_{ki}(x_i). \tag{10}$$

We could rewrite Eq. (9)in a form of 2D convolution, if $(i, j) \in \mathcal{E}$,

$$M_{ij} = \theta^{(i,j)} * H_i,\ M_{ji} = \left(\theta^{(i,j)}\right)^T * H_j, \tag{11}$$

where $M_{ij} \in \mathbb{R}^{h_u \times w_u}$, $\theta^{(i,j)} \in \mathbb{R}^{h_p \times w_p}$, $\theta \in \mathbb{R}^{|\mathcal{E}| \times h_p \times w_p}$, $H_i \in \mathbb{R}^{h_u \times w_u}$. The matrix H_i is the compact matrix formed by values of $h_i(x_i)$. Appropriate zero padding is required on H_i to make the shape of M_{ij} the same as that of H_i. This similar idea is also used in [22]. Kernel $\theta^{(i,j)}$ could be interpreted as the probability of where keypoint j would be with respect to keypoint i, and it encodes the information of relative positions between the keypoint i and j. In our implementation in Fig. 2, to avoid the transpose operation in Eq.(11), we let the pairwise branch produce an output $Q \in \mathbb{R}^{2|\mathcal{E}| \times h_p \times w_p}$ which has $2 \times |\mathcal{E}| = 40$ channels.

4 Learning

Since there are two branches of DCNN in the proposed AGMN, we utilize a 3-stage training strategy. Firstly, the unary branch is trained. Then, the pairwise branch is trained with the unary branch fixed. Finally, the whole AGMN is finetuned end-to-end.

Train unary branch. The unary branch is trained alone first. As in [19, 25], intermediate supervision is used during the training. Each stage of the unary branch is trained to repeatedly produce the score maps (or belief maps) for the locations of each of the hand keypoints. The ground truth score map of keypoint i, denoted as $S_i^* \in \mathbb{R}^{h_u \times w_u}$, is created by putting a Gaussian peak at its ground truth location. The cost function at each stage t of the unary branch is defined by

$$f_t = \sum_{k=1}^{21} ||S_k^t - S_k^*||_F^2, \tag{12}$$

where S_k^t is the score map of keypoint k generated by the t-th stage in the unary branch. Notation $||\cdot||_F$ represents the Frobenius norm which is defined as the square root of the sum of the squares of its elements. If we have T stages, then $S_k^T = \mathcal{U}^k(I; \Theta_u)$ in Eq.(2). By adding up the cost functions at each stage, the final loss function of the unary branch is

$$L^{unary} = \sum_{t=1}^{T} f_t. \tag{13}$$

Train pairwise branch. The pairwise branch is trained with the help of the unary branch, since there are some information flowing from the unary branch to the pairwise branch as shown in Fig. 2. Parameters of the unary branch are frozen during this training phase.

The goal of the pairwise branch is to learn relative positions between hand keypoints. The pairwise branch produces an output Q of 40 channels, with each channel corresponding to one directed edge in the message passing schedule. The ideal output (ground truth) Q^* of the pairwise branch is computed from relative positions of each pair of neighboring hand keypoints which share a common edge in the tree structure. For example, if the k-th directed edge (right side of Fig. 3) incidents on two hand keypoints i and j, say starting from i to j, the relative position of these two keypoints is computed as $r_k = l_j - l_i$, where l_i and l_j are length-2 vectors representing the ground truth positions of the keypoints. Then, the ground truth of the k-th channel of Q^*, i.e., Q^*_k, is created by putting a Gaussian peak at the location which is r_k away from the center of the 2D matrix.

We use a similar loss function as that in training the unary branch

$$L^{pairwise} = \sum_{t=1}^{T} \sum_{k=1}^{40} ||Q^t_k - Q^*_k||^2_F. \tag{14}$$

Fine tune the whole AGMN. Since the final outputs of the AGMN are marginal probabilities, the ground truth for the final score map of keypoint k, $\tilde{S}^*_k$ is set to be the normalized version of S^*_k used in Eq.(12). The loss function defined by the final score maps is given by

$$L^{last} = \sum_{k=1}^{21} ||\tilde{S}_k - \tilde{S}^*_k||^2_F, \tag{15}$$

where $\tilde{S}_k$ is the k-th channel of the output of the AGMN.

The whole AGMN is fine tuned with a loss function which is a weighted sum of loss functions from the unary branch, pariwise branch and module of graphical model as following

$$L = \alpha_1 L^{unary} + \alpha_2 L^{pairwise} + \alpha_3 L^{last}. \tag{16}$$

5 Experiments

In this section, we demonstrate the performance of our proposed algorithm on two real-world hand pose datasets. Comparative analysis is also carried out.

5.1 Experimental settings

Datasets. We evaluate our model on two public datasets, the CMU Panoptic Hand Dataset (referred to as "CMU Panoptic")[19], and the Large-scale Multiview 3D Hand Pose Dataset (referred to as "Large-scale 3D") [7]. (i) The CMU Panoptic dataset contains 14817 annotations of right hands in images of persons from Panoptic Studio. Since this paper's focus is on hand pose estimation other than hand detection, we cropped image patches of annotated hands off the original images using a square bounding box which is 2.2 times the size of the hand. Then, we randomly split the whole dataset into training set (80%), validation set (10%) and test set (10%). (ii) The Large-scale 3D dataset contains 82760 images in total. We follow the same preprocessing procedure on this dataset and take care of the keypoints ordering. Although this is a 3D dataset, it provides an interface to get 2D annotations by performing projection. The Large-scale 3D dataset is split into training set (60000 images), validation set (10000 images) and test set (12760 images).

Evaluation metric. We consider the "normalized" Probability of Correct Keypoint (PCK) metric from [19]: the probability that a predicted keypoint is within a distance threshold σ of its true location. We use a normalized threshold σ which ranges from 0 to 1, with respect to the size of hand bounding box.

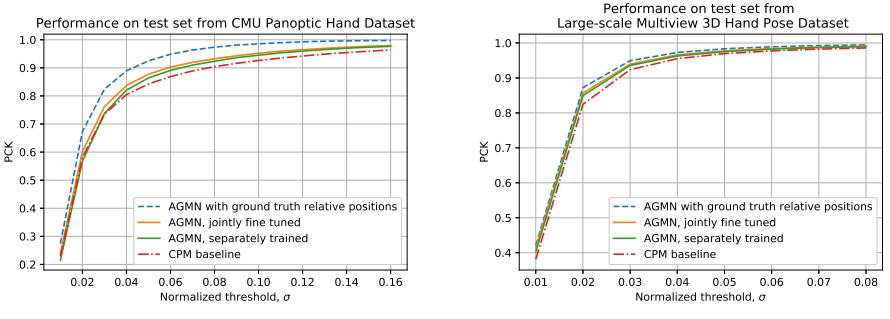
Implementation details. Our experiments are conducted using PyTorch. All images are resized to 368×368 before fed into the AGMN, yielding a final score map of size 46×46 for each keypoint. Also, after being scaled to [0,1], all the images are then normalized using mean = (0.485, 0.456, 0.406) and std = (0.229, 0.224, 0.225).

The output shape of the unary branch is designed to be $21 \times 46 \times 46$ as in [19], corresponding to 21 hand keypoints with heatmap resolution of 46×46. Meanwhile, we set the resolution of the output from the pairwise branch to be 45×45, by adding a downsampler after the VGG-19 feature extractor. With the resolution being an odd number, the central entry of the matrix corresponds to the case when the relative position between two hand keypoints is a zero vector.

During training, the batch size is set to 32. The `torch.nn.MSELoss(size_average=None, reduce=None, reduction='mean')` is utilized for the loss function. A coefficient of 30 is multiplied to this function to avoid the loss being too small. The gaussian peaks used to generate ground truth during training all have standard deviation of 1. Learning rate is set to 1e-4 when training the unary branch and pairwise branch. When finetuning the whole AGMN, learning rate is set to 1e-5 and the coefficients in Eq.(16) are set to $\alpha_1 = 1, \alpha_2 = 0.1, \alpha_3 = 0.1$.

5.2 Results

Fig. 4 shows our model's performance on above mentioned datasets. Detailed numerical results are summarized in Table. 1. It is seen that our model outperforms CPM consistently.



(a) CMU Panoptic. (b) Large-scale 3D Dataset.

Figure 4: Model performance.

On CMU Panoptic dataset, by training the unary branch and pairwise branch separately, we see an absolute PCK improvement of 2.12% at threshold $\sigma = 0.05$. A final improvement of 3.45% is obtained after finetuning the unified AGMN. On Large-scale 3D dataset, our AGMN obtains its highest improvement 3.27% at thresholds $\sigma = 0.01$. The authors in [22] stated that "Spatial-Model has little impact on accuracy for low radii threshold". However, based on the results in Fig. 4(b), it is observed that our *adapitve* spatial model has the power of increasing accuracy for low radii threshold.

The reason why AGMN achieves highest improvement on CMU Panoptic dataset at higher threshold σ than that of Large-scale 3D dataset, probably lies in the fact that CMU Panoptic dataset is a much harder dataset where a lot more occlusions exist.

We also conducted an experiment where the ground truth of the relative positions among hand keypoints (Q^* in Eq.(14)) are given to the AGMN, with pre-trained unary branch. The result of this experiment is actually the upper bound of our AGMN's performance given specific unary branch. The result is drawn as the blue dashed lines in Fig. 4.

Threshold of PCK, σ	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
CMU Panoptic Hand Dataset										
CPM Baseline (%)	22.88	58.10	73.48	80.45	84.27	86.88	88.91	90.42	91.61	92.61
AGMN Sep. Trained	21.52	56.73	73.75	82.06	86.39	89.10	91.00	92.35	93.63	94.50
AGMN Fine Tuned	23.90	60.26	76.21	83.70	87.72	90.27	91.97	93.23	94.30	95.20
Improvement	1.02	2.16	2.73	**3.25**	**3.45**	**3.39**	**3.06**	2.81	2.69	2.59
Large-scale Multiview 3D Hand Pose Dataset										
CPM Baseline (%)	38.11	82.48	92.37	95.50	96.97	97.75	98.24	98.58	98.84	99.02
AGMN Sep. Trained	40.22	84.94	93.57	96.29	97.53	98.24	98.68	98.97	99.17	99.34
AGMN Fine Tuned	41.38	85.67	93.96	96.61	97.77	98.42	98.82	99.10	99.29	99.43
Improvement	**3.27**	**3.19**	1.59	1.11	0.80	0.67	0.58	0.52	0.45	0.41

Table 1: Detailed numerical results.

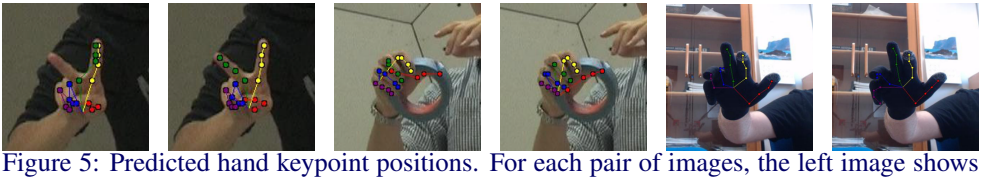


Figure 5: Predicted hand keypoint positions. For each pair of images, the left image shows the result of CPM and the right image shows that of AGMN.

Examples in Fig. 5 show that our AGMN could greatly reinforce the keypoints consistency and reduce ambiguities in prediction. Note that the first keypoint in Large-scale 3D dataset is the center of the palm instead of the wrist.

6 Conclusion

This paper provides a new direction on how deep convolutional neural networks can be combined and integrated with graphical models. We propose an adaptive framework called AGMN for hand pose estimation, which contains two branches of DCNN, one for regressing the score maps of hand keypoint positions, the other for regressing the parameters of graphical model, followed by a graphical model for inferring the final score maps through message passing. The novelty of our AGMN lies in that the parameters of graphical model are fully adaptive to individual input images, instead of being shared by input images. Experiment results show that the proposed AGMN outperforms the commonly used CPM algorithm on two public hand pose datasets. The proposed framework is general and can also be applied to other deep learning applications where performance can benefit by considering structural constraints.

References

[1] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Augmented skeleton space transfer for depth-based hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8330–8339, 2018.

[2] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3D hand pose estimation from monocular RGB images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–682, 2018.

[3] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[4] Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun. Learning deep structured models. In *International Conference on Machine Learning*, pages 1785–1794, 2015.

[5] Xianjie Chen and Alan L Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in neural information processing systems*, pages 1736–1744, 2014.

[6] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: regional multi-person pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2334–2343, 2017.

[7] Sergio Orts-Escolano Francisco Gomez-Donoso and Miguel Cazorla. Large-scale multiview 3d hand pose dataset. *ArXiv e-prints 1707.03742*, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[10] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. DeeperCut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.

[11] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov, Bjoern Andres, and Bernt Schiele. ArtTrack: Articulated multi-person tracking in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6457–6465, 2017.

[12] Umar Iqbal, Anton Milan, and Juergen Gall. PoseTrack: Joint multi-person pose estimation and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2020, 2017.

[13] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart Nabbe, Iain Matthews, et al. Panoptic Studio: A massively multiview system for social interaction capture. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):190–204, 2019.

[14] Lipeng Ke, Ming-Ching Chang, Honggang Qi, and Siwei Lyu. Multi-scale structure-aware network for human pose estimation. In *European Conference on Computer Vision*. Springer, 2018.

[15] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked Hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[16] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single RGB frame for real time 3D hand pose estimation in the wild. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 436–445. IEEE, 2018.

[17] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4903–4911, 2017.

[18] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. DeepCut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937, 2016.

[19] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[21] Jie Song, Limin Wang, Luc Van Gool, and Otmar Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[22] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[23] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dense 3D regression for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2018.

[24] Zhe Wang, Liyan Chen, Shaurya Rathore, Daeyun Shin, and Charless Fowlkes. Geometric pose affordance: 3d human pose with scene constraints. In *Arxiv*, page 1905.07718, 2019.

[25] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[26] Wei Yang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3073–3082, 2016.

[27] Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, et al. Depth-based 3D hand pose estimation: From current achievements to future goals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2018.

[28] Christian Zimmermann and Thomas Brox. Learning to estimate 3D hand pose from single RGB images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4903–4911, 2017.