Adapting RGB Pose Estimation to New Domains

Gururaj Mulay, Bruce A. Draper, J. Ross Beveridge
Department of Computer Science
Colorado State University
Fort Collins, CO USA
guru5@colostate.edu, draper@colostate.edu, ross@colostate.edu

***Abstract*—Many multi-modal human computer interaction (HCI) systems interact with users in real-time by estimating the user's pose. Generally, they estimate human poses using depth sensors such as the Microsoft Kinect. For multi-modal HCI interfaces to gain traction in the real world, however, it would be better for pose estimation to be based on data from RGB cameras, which are more common and less expensive than depth sensors. This has motivated research into pose estimation from RGB images. Convolutional Neural Networks (CNNs) represent the state-of-the-art in this literature, for example [1], [2], [9], [13], [14], and [15]. These systems estimate 2D human poses from RGB images. A problem with current CNN-based pose estimators is that they require large amounts of labeled data for training. If the goal is to train an RGB pose estimator for a new domain, the cost of collecting and more importantly labeling data can be prohibitive. A common solution is to train on publicly available pose data sets, but then the trained system is not tailored to the domain. We propose using RGB+D sensors to collect domain-specific data in the lab, and then training the RGB pose estimator using skeletons automatically extracted from the RGB+D data. This paper presents a case study of adapting the RMPE pose estimation network [2] to the domain of the DARPA Communicating with Computers (CWC) program [3], as represented by the EGGNOG data set [8]. We chose RMPE because it predicts both joint locations and Part Affinity Fields (PAFs) in real-time. Our adaptation of RMPE trained on automatically-labeled data outperforms the original RMPE on the EGGNOG data set.**

***Index Terms*—HCI, human pose estimation, Microsoft Kinect, Communicating with Computers**

I. Introduction

Many multi-modal human computer interaction (HCI) systems interact with users in real-time in part by estimating the user's pose. For example, Narayana *et al.* [3] describe a multi-modal interface for an avatar, in which users gesture and/or speak to direct an avatar. As shown in Figure 1, the virtual agent avatar perceives the user's motions by estimating their pose over time. The agent analyzes the predicted joint locations (marked in yellow), and matches them to known gestures (or to *no gesture*) and responds appropriately. In this way, pose estimation facilitates gesture comprehension, which in turn helps the system achieve multi-modal communication via gestures and speech.

This work was supported by DARPA and ARO through grant W911NF-15-1-0459.

978-1-7281-0554-3/19/$31.00 ©2019 IEEE

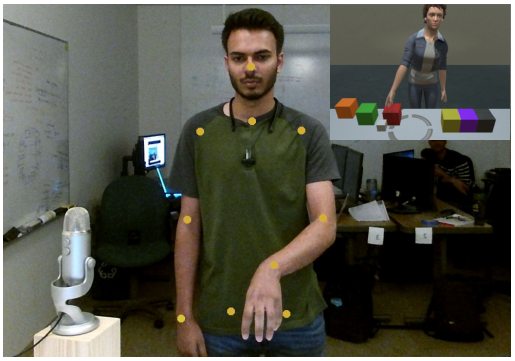

Fig. 1. Communicating with Computers program: multi-modal communication between user and agent (inset)

The system described by Narayana *et al.* exploits body poses (a.k.a. skeletons) produced by the Microsoft Kinect sensor. This limits the domains it can be applied to, since most users do not own Kinects. The next step is to develop a multi-modal interface that extracts pose information from RGB cameras, which are already cheap and ubiquitous.

The state of the art technique for pose estimation from RGB images is the convolutional neural networks; examples include [1], [2], [9], [13], [14], and [15]. In particular we focus on RMPE [2], which is widely cited and real-time. RMPE is a multi-stage CNN that extracts the image positions of body joints (elbows, wrists, etc.) as well as part affinity fields representing the limbs that connect joints (e.g. the arm between the elbow and wrist).

One challenge in adapting RMPE (or any CNN) to a new domain is collecting the training data. Generally, pose estimator such as RMPE are trained on manually annotated data sets such as COCO [5], [6], and [7]. Manual labeling is an expensive process, however, and prone to errors and inconsistencies in label definitions. An alternative is to collect training images with co-registered depth images (e.g. with Microsoft Kinect) and use existing pose-from-depth algorithms to automatically label the joint positions. This approach is inexpensive and quick, and makes it easy to collect large amounts of labeled training images with consistent label definitions. However, it potentially introduces errors into the training data, since the pose-from-depth algorithms make mistakes and depth data can be noisy. It is an open question how well RMPE will perform

when trained on automatically-extracted skeletons.

Figure 2 illustrates the adaptation scenario. The idea is to collect data with a Kinect in the lab (as shown in the upper left) and to train RMPE to label RGB images (bottom left) based on automatically-generated skeletons. Once the network is trained, it can be used outside the lab to estimate poses from traditional RGB cameras (right side of Figure 2). The final application therefore only requires a laptop and its built-in camera.

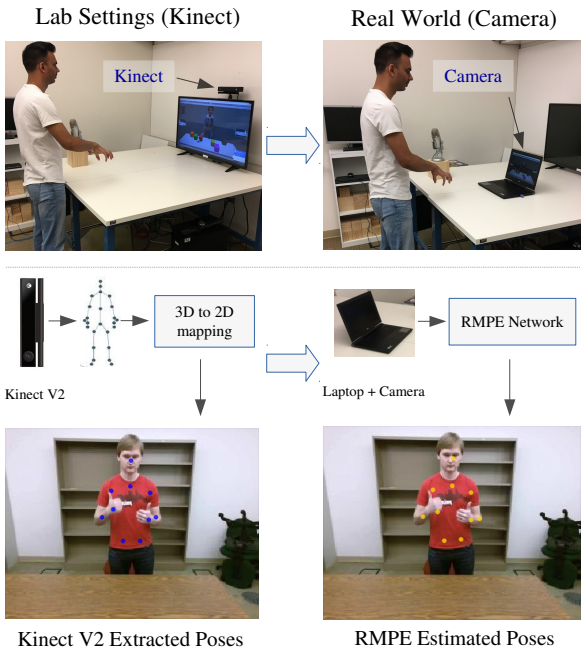

Fig. 2. Lab setup uses the Kinect v2 for training versus the real world application that uses a traditional RGB camera.

This paper is therefore a case study in adapting RMPE to new domains. In this case, the domain is multi-modal HCI, as represented by the EGGNOG [8] data set. The first question is whether RMPE can be retrained with minimal human effort by collecting automatically-labeled data from a Microsoft Kinect v2, and if it can whether the result will outperform the standard, off-the-shelf RMPE trained on COCO [5]. Since the answer to this question turns out to be yes, RMPE can be retrained using automatically-labeled data and doing so does improve performance on the target domain, the remaining questions concern how best to configure RMPE for the said domain. In particular, we investigate (1) the use of spatial dropout, (2) how many stages are necessary, (3) how many training images are needed and (4) how many training subjects are needed.

II. Literature Review

Pose estimation from RGB images is an active area of research with numerous practical applications in areas such as HCI, motion capture, and augmented reality [3], [24], and [25]. Prior to the advent of CNNs, pose estimation techniques were based on pictorial structures [16]–[19] and graphical models [10], [20]. These methods predicted joint locations from hand-crafted features. Recently, CNN-based methods have been shown to regularly outperform these classical methods by large margins [1], [2], [9], [11]–[13]. Toshev *et al.* [9], formulated pose estimation as regression to the coordinates of joints using a generic CNN. Joint relations were learned instead of designed by hand, making the network generalizable. Building on this regression concept, Tompson *et al.* [4], [21] and Newell *et al.* [13] formulated CNNs that regress input images to confidence maps depicting the probabilities of the presence of joints. Wei *et al.* [1] used a multi-stage CNN to regress with larger receptive fields, allowing their network to learn stronger spatial dependencies over successive stages. More recently, Chen *et al.* [14] introduced a two-staged architecture consisting of GlobalNet and RefineNet. Xiao *et al.* [15] present a simple and effective architecture based on a ResNet [23] backbone network with the addition of deconvolution layers to predict confidence maps. All of these networks are based on the fundamental idea of regressing images to confidence maps.

The Convolutional Pose Machine (CPM) [1] and RMPE [2] have multi-staged CNNs that regress images to confidence maps depicting joint locations. The predictions of these networks are refined over successive stages as their receptive fields increase in deeper stages. Recent work [1], [2], [13], [14], and [21] suggests that multi-stage CNNs learn more expressive features and implicit spatial dependencies between joints directly from large-scale data and perform better when compared to classical methods. CPM employs a top-down approach wherein a person detector outputs a detection that is fed to a single-person pose estimation network. The runtime of the top-down approach is proportional to the number of people in the image. In contrast, RMPE employs a bottom-up approach wherein a single network detects and estimates joints for all the people in the image. RMPE uses 'simultaneous detection and association.' The network predicts confidence maps for joint locations followed by associating PAFs that encode part-to-part relations. Cao *et al.* show that the runtime of RMPE with its bottom-up approach increases relatively slowly with respect to the number of persons in the image.

In this paper, we adopt the common pose estimation formulation which regresses image to confidence maps. In particular, we concentrate directly on the pose machines published by CMU (CPM [1] and RMPE [2]) that won the COCO 2016 Keypoints Challenge. We analyze RMPE on a new large-scale data set (EGGNOG [8]). We contribute by retraining and evaluating RMPE on EGGNOG data set and by providing an analysis of the RMPE adaptation process to a specific domain.

III. Adapting and Retraining RMPE

Our goal is to determine whether RMPE can be retrained for a specific domain without manually labeling data, and if doing so produces a network that outperforms the standard RMPE trained on a general-purpose data set. This section describes how RMPE is adapted and retrained for the EGGNOG domain.

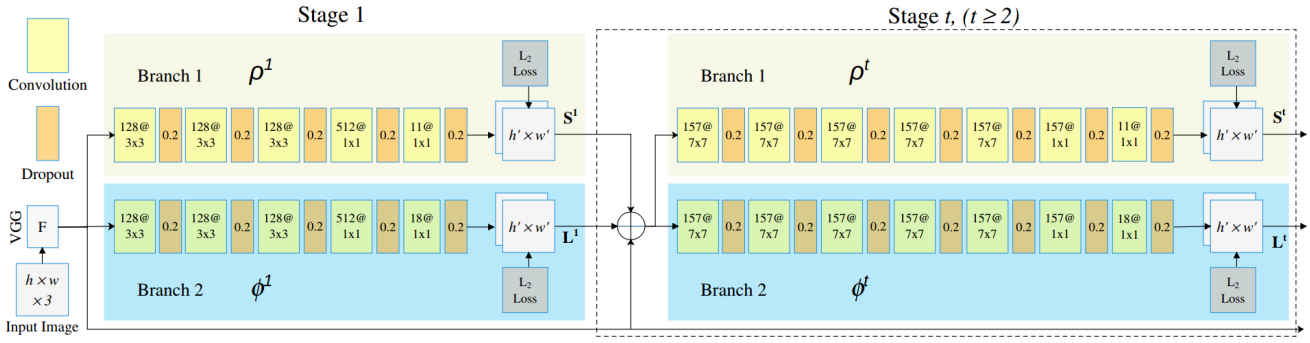

Fig. 3. General architecture of our adapted RMPE for CWC domain: two-staged RMPE inspired by [2]

Later, it compares the relative performance of the original and retrained versions of RMPE.

A. Adapting RMPE for EGGNOG

Ideally, we would retrain RMPE without changing its architecture in any way, to create to perfect apples-to-apples comparison in subsection *C*. Unfortunately, differences between the COCO and EGGNOG data sets require small architectural changes. Users in EGGNOG are standing behind a table; their legs and feet are not visible. We therefore train RMPE to detect the 10 upper body joints that are common to the Microsoft skeleton and the COCO data labels, meaning that our adapted RMPE predicts 11 confidence maps (10 for joints and one for background) and 18 PAF maps for connectors formed by the 10 joints. In contrast, standard RMPE predicts 19 confidence maps and 38 PAF maps based on the manually annotated labels in the COCO data set. Therefore we change the final convolutional layer at every stage of RMPE to predict 11 feature maps and 18 PAF maps. We also added spatial dropout layers between convolutional layers to avoid over-fitting, and limited the depth of the architecture to two stages, for reasons that will be explained in Section IV.

Figure 3 shows our modified RMPE. The details are similar to those in [2], but are included here for completeness. RGB images are fed through the VGG feature extractor block [22] to generate feature maps $\mathbf{F}$. These feature maps ($\mathbf{F}$) are fed as input to both the branches of stage 1. The first stage outputs confidence maps $\mathbf{S}^1 = \rho^1(\mathbf{F})$ and PAF maps $\mathbf{L}^1 = \phi^1(\mathbf{F})$ where ρ^1 and ϕ^1 are the CNNs from branch 1 and 2 of the first stage. The structure of convolutional block for every stage after stage 2 is identical to stage 2 block structure. For each stage after stage 1, the predictions from both the branches of the previous stage and VGG feature maps $\mathbf{F}$ are concatenated and fed as input to the next stage such that,

$$\mathbf{S}^t = \rho^t(\mathbf{F}, \mathbf{S}^{t-1}, \mathbf{L}^{t-1}), \forall t \geq 2, \tag{1}$$

$$\mathbf{L}^t = \phi^t(\mathbf{F}, \mathbf{S}^{t-1}, \mathbf{L}^{t-1}), \forall t \geq 2, \tag{2}$$

where, ρ^t and ϕ^t are the CNNs from branch 1 and 2 of stage t. This formulation is similar to what is proposed in [2].

Replicating the loss functions from the original RMPE, we use L_2 loss function at the end of each stage to enforce intermediate supervision. For each stage, the losses are calculated at the end of each branch between predicted feature maps and ground truth feature maps. For stage t the losses are calculated as follows,

$$f_{\mathbf{S}}^t = \sum_{j=1}^{J} \|\mathbf{S}_j^t - \mathbf{S}_j^*\|_2^2, \tag{3}$$

$$f_{\mathbf{L}}^t = \sum_{c=1}^{C} \|\mathbf{L}_c^t - \mathbf{L}_c^*\|_2^2, \tag{4}$$

where $\mathbf{S}_j^*$ and $\mathbf{L}_c^*$ are the ground truth confidence maps and PAF maps respectively. Process to generate ground truths for EGGNOG is discussed in the next subsection. Total loss f of the network is sum of all the losses at each stage and each branch defined by,

$$f = \sum_{t=1}^{T} (f_{\mathbf{S}}^t + f_{\mathbf{L}}^t), \tag{5}$$

where, $f_{\mathbf{S}}^t$ and $f_{\mathbf{L}}^t$ are the losses at stage t for the confidence map predictor and PAF predictor branches respectively.

B. Experimental Methodology

The EGGNOG [8] data set has approximately 300K RGB image frames in 360 trials of 40 human subjects, for a total of 8 hours of data. It was recorded using a Microsoft Kinect v2 sensor, and includes 3D skeletons for 25 joints (and 2D projections thereof) extracted from the RGB+D data. The experiments conducted in this thesis use 10 joints: head, spine shoulder, left shoulder, left elbow, left wrist, right shoulder, right elbow, right wrist, left hip, and right hip. We divide the EGGNOG data set in training, validation, and testing sets containing 28, 8, and 4 subjects respectively.

The EGGNOG image frames are 1920 x 1080 pixels. We remove 240 pixel columns from the left and right sides, since EGGNOG subjects stay near the center of the frames. We reduce the image scale by a factor of 4.5 to get a 320×240 image. This size is close to what the original RMPE uses (368×368).

We use the algorithms specified in [2] to generate the ground truth confidence maps $\mathbf{S}^*$ and PAF maps $\mathbf{L}^*$. The confidence maps are Gaussian in 2D pixel space representing the belief that a joint occurs at a specific pixel location. To generate the confidence map for joint *j*, we use the 2D joint annotations ($\mathbf{x}_j \in \mathbb{R}^2$) from Kinect data after transforming them to a 40 x 30 ground truth space. The first row of Figure 5 shows the confidence maps for right elbow and right wrist overlaid on down-sampled input image. Confidence map values range from 0 to 1 with 0 meaning no belief and 1 meaning complete belief that a particular joint is present at that pixel. The value of confidence map for joint *j* at pixel location $\mathbf{p} \in \mathbb{R}^2$ is defined by,

$$\mathbf{S}_j^*(\mathbf{p}) = \mathbf{exp}(-\alpha \times \|\mathbf{p} - \mathbf{x}_j\|_2^2), \tag{6}$$

where, α determines the spread of the Gaussian belief map.

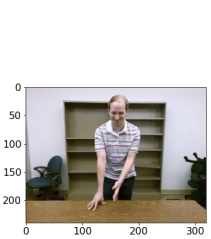

Fig. 4. Input RGB image (320 × 240 × 3)

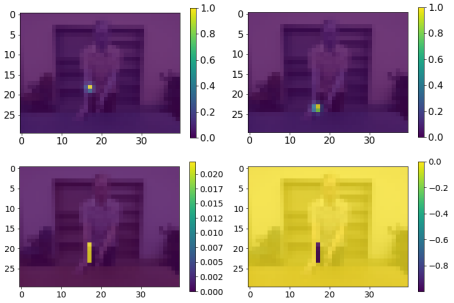

Fig. 5. Ground truth (40×30): 1^{st} row - confidence maps for left (since images flipped horizontally) elbow and left wrist; 2^{nd} row - PAF maps (x and y) for joint connector from left elbow to left wrist

PAF maps are generated for all the joint connectors *c* from the set of joint connectors *C*. Consider a joint connector *c* formed by connecting joint j_1 at location $\mathbf{x}_{j_1}$ to joint j_2 at location $\mathbf{x}_{j_2}$. The value of PAF map for a joint connector *c* at pixel location $\mathbf{p} \in \mathbb{R}^2$ is defined by,

$$\mathbf{L}_c^*(\mathbf{p}) = \begin{cases} \mathbf{v} & \text{if } \mathbf{p} \text{ is on joint connector } c. \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{7}$$

Here, $\mathbf{v}$ is a unit vector from joint j_1 to j_2 defined by,

$$\mathbf{v} = (\mathbf{x}_{j_2} - \mathbf{x}_{j_1})/\|\mathbf{x}_{j_2} - \mathbf{x}_{j_1}\|_2, \tag{8}$$

A point $\mathbf{p}$ is considered to be on the joint connector *c* if it follows

$$0 \leq \mathbf{v} \cdot (\mathbf{p} - \mathbf{x}_{j_1}) \leq l_c \text{ and } |\mathbf{v}_\perp \cdot (\mathbf{p} - \mathbf{x}_{j_1})| \leq \sigma_l. \tag{9}$$

Here, $l_c = \|\mathbf{x}_{j_2} - \mathbf{x}_{j_1}\|_2$ is the length of the joint connector from joint j_1 to j_2 and σ_l is the width of the same connector in pixels.

We augment the RGB images by randomly choosing an angle of rotation from [-12°, +12°], scaling factor from [0.8, 1.2], horizontal flipping probability of 0.5, and translation value (in number of pixels) along horizontal and vertical direction from [-40, 40]. While generating the joint confidence maps and PAF maps for EGGNOG, we selected α = 2.25 and limb width (σ_l) = 0.75 in equation 6 and 7 respectively.

The network is trained with the same parameters as in [2]. The base learning rate is 4×10^{-5}, the momentum factor for Stochastic Gradient Descent (SGD) optimizer is 0.9, and the weight decay factor is 5×10^{-4}. For the EGGNOG data set the network converges after approximately 100 epochs.

Experiments are evaluated with Percentage of Correct Keypoints (PCK) metric. For PCK, a keypoint is considered to be correct if its predicted location falls within a normalized distance of its ground truth location. As proposed in [10], the normalized distance is defined using the formula $\alpha \cdot \max(h, w)$, where h and w are respectively the height and width of the tightest bounding box that encloses all the ground truth keypoints of the test sample. We vary α from 0 to 0.5 because for $\alpha > 0.5$ the PCK vs. normalized distance plot saturates to PCK $\approx$ 1.0.

IV. Evaluating RMPE Trained on Automatically-Labelled Data

The primary goal of this experiment is to test the feasibility of re-training RMPE using automatically-extracted skeleton data. Hand-labeling RGB images is expensive; if we can collect RGB+D images and use the automatically-extracted skeletons as training data, the labor cost of retraining RMPE drops significantly. The risk, however, is that errors in the training signal could degrade RMPE's performance. To test this, we compare the retrained RMPE's performance to the performance of the original RMPE trained on a broader domain but with hand-labeled data.

The network for this experiment shown in Fig. 3 consists of a two-staged RMPE network predicting both the joint locations and PAFs. Each stage iteratively refines the predicted joint locations. The training, validation and test sets are drawn from the EGGNOG data set [8]. The training set consists of 40K images evenly distributed among 28 subjects while the validation set contains 4K images evenly distributed among different 8 subjects. We reserve 4 subjects for the test set. The modified RMPE network predicts confidence maps for 10 joints listed in the experimental methodology subsection.

We compare the performance of the original RMPE [2] and our retrained model using two metrics, the Percentage of Correct Keypoints at a normalized distance of 0.1 (PCK@0.1) and the Area Under the Curve (AUC) calculated for the 'Mean PCK versus Normalized distance' plot.

Figure 6 compares the performance of the original and automatically-trained systems by plotting the average percent of correct keypoints (PCK) as a function of the threshold used to determine correctness. The threshold is a percentage of the shorter side of the bounding rectangle that contains the subject. At large values of the threshold (above 0.3, or 30% of the bounding window size), both systems get 100% of the keypoints correct. This is because the threshold is so lenient that joint positions almost anywhere near the body qualify as correct. At lower thresholds, however, the automatically-trained version of RMPE (shown in blue) consistently outper-

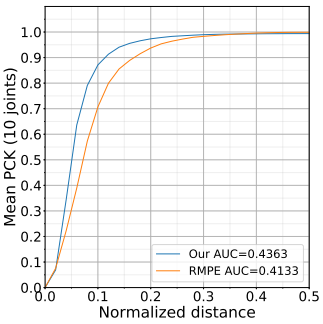

Fig. 6. Mean PCK of 10 joints: original RMPE tested on EGGNOG vs. our adapted RMPE trained and tested on EGGNOG

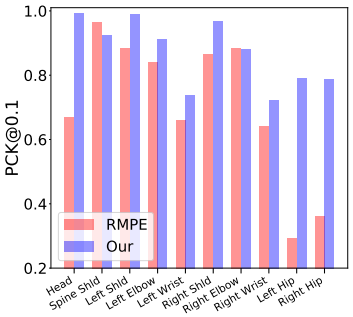

Fig. 7. PCK@0.1 for individual joints: original RMPE tested on EGGNOG vs. our adapted RMPE trained and tested on EGGNOG

forms the original (shown in red). Since the horizontal axis ranges from 0 to 0.5, the graph of a perfect system would yield an area under the curve (AUC) of 0.5. The retrained RMPE has an AUC of 0.4363, whereas the original RMPE has an AUC of only 0.4133.

Figure 7 compares the two systems on a joint-by-joint basis, at a fixed distance threshold of 0.1. We see that our retrained version outperforms the original for 8 out of the 10 joints. The performance of two systems are essentially tied for one joint (right elbow) and the original RMPE is best for 1 joint (spine shoulder). The retrained system is therefore performing broadly better. The difference in performance is particularly noticeable for the hips; this may be because of inconsistency in how hips were labeled in COCO, or because of a definitional mismatch as to where the hips are.

V. Additional Experiments and Evaluations

Having shown that RMPE can be trained using automatically-generated skeletons, we needed to find the best parameter configuration for retraining RMPE on a tightly controlled data set. In particular, we looked at spatial dropout, the number of RMPE network stages, the number of training subjects, and the number of training samples.

A. Experiments with Spatial Dropout

The original version of RMPE was trained without drop-outs. We added drop-outs when we modified RMPE to avoid over-fitting. We observed in early experiments with our system that adding spatial dropout layers with a dropout rate of 0.2 after each convolutional layer reduced over-fitting on EGGNOG. The experiments in this section analyze how different dropout strategies affect network performance.

We use spatial dropout regularization formulated recently by Tompson *et al.* [4] to improve network performance while solving pose estimation. They argue that standard dropout techniques do not prevent over-fitting because the pixels within a feature map are so strongly correlated. To avoid this problem they introduce spatial dropout where activations of entire feature map are randomly set to zero. Similar to their results, we observe improvements in performance of our network with addition of spatial dropout. For these experiments, we apply spatial dropout with a certain probability after every convolutional layer in the stages of RMPE (shown in Fig. 3) to search for the best dropout strategy in the context of EGGNOG.

Figure 8 and Figure 9 show the mean PCK scores for five trials (plotted with red dots) of experiments with various spatial dropout strategies. With a spatial dropout rate of 0.0, the AUC for PCK@0.1 is 0.8462. It increases to 0.8786 with a dropout rate of 0.2. If we further increase the dropout rate to 0.3, the mean PCK@0.1 drops down to 0.8544. We experimented with other variations of dropout and concluded that for the EGGNOG data a dropout of 0.2 along the layers gave the best performance.

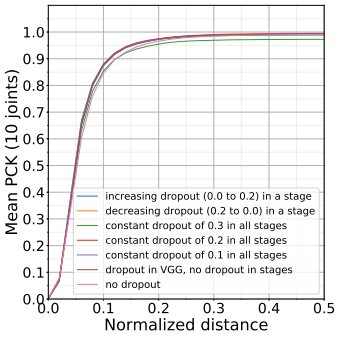

Fig. 8. Mean PCK of 10 joints w.r.t. various spatial dropout schemes

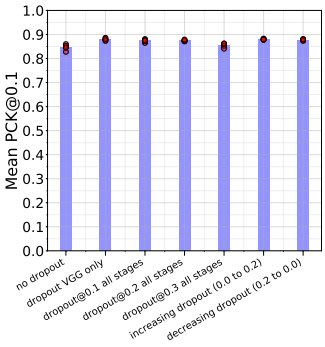

Fig. 9. Mean PCK@0.1 for various spatial dropout schemes

B. Experiments with the Number of Stages

RMPE is a multi-stage architecture, where each stage refines the results of the preceding stage. This experiment seeks to understand how many stages are needed in the context of EGGNOG. Cao *et al.* in [2] report that RMPE performance improves monotonically with the number of stages, but that the improvement going from a one-staged network to a two-staged network is bigger than the improvements gained by adding additional layers beyond the second.

Using the same training set, network parameters, and learning parameters as described above, this experiment changes the number of stages and analyzes the effect on PCK scores. For each experiment we run five trials to get central tendency of PCK scores.

Fig. 10 shows mean PCK for 10 joints against the normalized distance for six experiments with number of stages varying from 1 to 6. Fig. 11 shows mean PCK@0.1 with five trials of each experiment.

The mean PCK score for one-staged networks is lower than the PCK scores for networks with more than one stage. Mean PCK@0.1 for one-staged network is 0.8320 (mean over five trials). This score increases to 0.8745 for two-staged network. However, there is no significant improvement in performance when additional stages beyond the second are added, unlike what was reported in the original RMPE paper. We analyzed the training times of these experiments on Nvidia TITAN V 12GB GPU. The average training epoch time for a one-staged

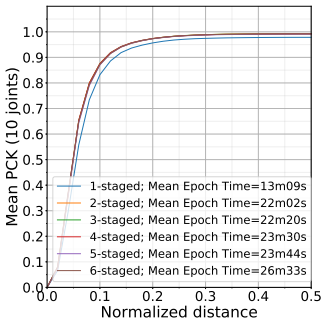

Fig. 10. Mean PCK of 10 joints w.r.t. number of stages

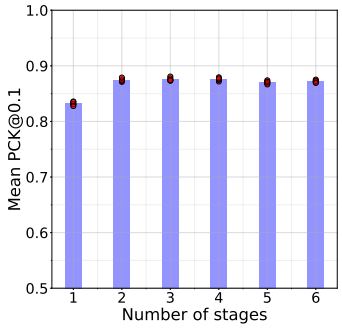

Fig. 11. Mean PCK@0.1 w.r.t. number of stages

network is 13:09 minutes. Adding one more stage increases the average epoch time to 22:02 minutes. As we increase the number of stages to six, we observe that the addition of each stage increases the training time by only a small amount. suggesting that the time increases are sub-linear.

We conclude that for the EGGNOG data set, a two-staged network is a good choice because 1) it has a better PCK@0.1 score than a one-staged network and 2) two-stage networks have shorter training times and fewer convolutional layers than networks with more stages, although the increase in training time is not as large as we originally expected.

C. Experiments with the Number of Training Samples

These experiments study how the of number of EGGNOG training samples affects PCK performance. Cao *et al.* trained RMPE [2] on the COCO data set which contains over 100K hand-labeled images. EGGNOG has approximately 300K instances of automatically annotated images. We study how many of these images are actually needed to retrain RMPE.

Using the same network and learning parameters as in the previous experiments, we varied the number of training samples. In particular, the distribution of training and validation samples for conducted experiments was 650:63, 1.25K:125, 2.5K:250, 5K:500, 10K:1K, 20K:2K, 40K:4K, and 80K:8K. For each experiment we ran five trials to get a central tendency of PCK scores.

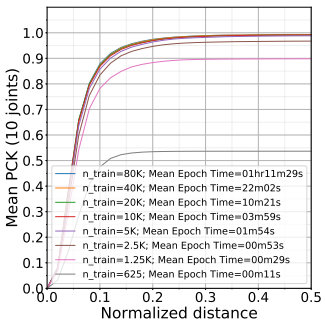

Fig. 12. Mean PCK of 10 joints w.r.t. number of training samples

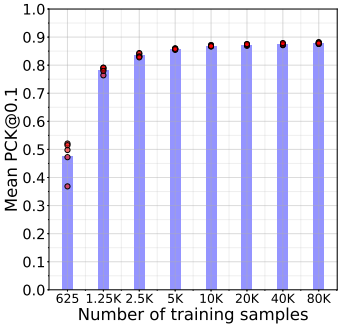

Fig. 13. Mean PCK@0.1 w.r.t. number of training samples

Fig. 10 shows the mean PCK for 10 joints against the normalized distance for the experiments with varying number of training samples. Fig. 11 shows mean PCK@0.1 with five trials of each experiment. With 40K training images, we get a mean PCK@0.1 of 0.8745. With 80K training images, the mean PCK@0.1 improves slightly to 0.8784. This suggests that there are only marginal benefits to using the entire EGGNOG data set for training. However, the mean PCK@0.1 deteriorates rapidly if the number of training samples is below 2K.

Unsurprisingly, training times increase as the number of training samples is increased. Average epoch times are in the orders of a few minutes for training sets with less than 10K images and in the order of hours for training sets with greater than 80K images. Our experiments suggest that for EGGNOG, the training set should be from 20K to 40K for a two-staged network.

D. Experiments with the Number of Training Subjects

The number of training samples is one measure of the size of a training set; the number of test subjects is another. In general, it is easier to collect more images per subject than to get more subjects. These experiments address the question of how many training subjects are needed to make RMPE perform well on EGGNOG. EGGNOG overall has 40 subjects, divided into the train, validation, and test sets. These experiments measure how many subjects are needed.

We vary the number of training subjects from 4 to 28. For each number of subjects we conducted five trials, with each trial having a different subset of training and validation subjects.

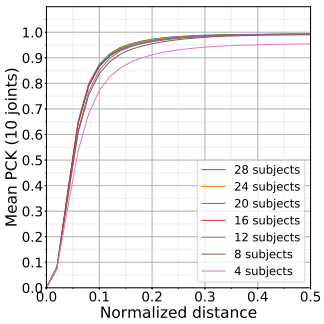

Fig. 14. Mean PCK of 10 joints w.r.t. number subjects in training set

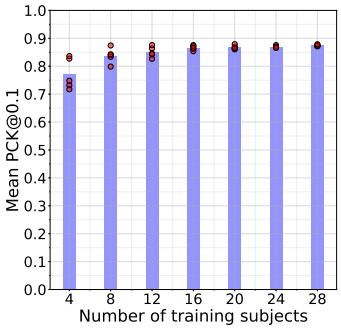

Fig. 15. Mean PCK@0.1 w.r.t. number of subjects in training set

Figure 14 shows the mean PCK for 10 joints while varying number of training subjects. Figure 15 is mean PCK@0.1 for 10 joints with five trials of each experiment.

We see in Figure 14 and Figure 15 that performance degrades as the number of training subjects is decreased. PCK@0.1 is 0.7722 with 4 training subjects and 0.8745 with 28 training subjects, although the difference between 16 and 28 subjects is small. We argue that with fewer training subjects the network may over-fit to the physical appearance of training subjects and therefore not generalize as well to the test set. We conclude that for EGGNOG the network should be trained on at least 16 subjects.

VI. Conclusion and Discussion

We present a case study on adapting and retraining a popular CNN-based pose estimator (RMPE [2]) to the EGGNOG data set. By tailoring RMPE to the HCI domain represented by the EGGNOG data set, we get better performance than if we use the pre-trained (on COCO) version of RMPE. Experiments also reveal that the EGGNOG domain only needs a two-staged RMPE to achieve strong performance, and the addition of more stages increases overhead without significantly improving performance. This paper shows too that spatial dropout regularization improves performance, even though it was not used in the original version of RMPE. Further, our analysis of retrained RMPE on EGGNOG shows that the training set should contain at least 16 human subjects and 40K training images in order to generalize.

Acknowledgment

This work was supported by the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) under contract #W911NF-15-1-0459 at Colorado State University.

References

[1] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. *CoRR*, abs/1602.00134, 2016.

[2] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016.

[3] P. Narayana, N. Krishnaswamy, I. Wang, R. Bangar, D. Patil, G. Mulay, K. Rim, R. Beveridge, J. Ruiz, J. Pustejovsky, and B. Draper. Cooperating with avatars through gesture, language and action. *Intelligent Systems Conference (IntelliSys)*, 2018.

[4] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014.

[5] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[6] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. 2014.

[7] M. Andriluka, U. Iqbal, A. Milan, E. Insafutdinov, L. Pishchulin, J. Gall, and B. Schiele. Posetrack: A benchmark for human pose estimation and tracking. *CoRR*, abs/1710.10000, 2017.

[8] I. Wang, M. Fraj, P. Narayana, D. Patil, G. Mulay, R. Bangar, R. Beveridge, B. Draper, and J. Ruiz. EGGNOG: A continuous, multi-modal data set of naturally occurring gestures with ground truth labels. In *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, pages 414–421, May 2017.

[9] A. Toshev and C. Szegedy. DeepPose: Human Pose Estimation via Deep Neural Networks. *CoRR*, abs/1312.4659, 2013.

[10] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, Dec 2013.

[11] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, Pe. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. *CoRR*, abs/1511.06645, 2015.

[12] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. *CoRR*, abs/1605.03170, 2016.

[13] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016.

[14] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded pyramid network for multi-person pose estimation. *CoRR*, abs/1711.07319, 2017.

[15] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking. *CoRR*, abs/1804.06208, 2018.

[16] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22(1):67–92, Jan 1973.

[17] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. *CVPR*, 2009.

[18] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. *CVPR*, pages 623–630, 2010.

[19] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Poselet conditioned pictorial structures. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 588–595, Washington, DC, USA, 2013. IEEE Computer Society.

[20] X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. *CoRR*, abs/1407.3399, 2014.

[21] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1799–1807. Curran Associates, Inc., 2014.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

[24] I. Wang, P. Narayana, D. Patil, G. Mulay, R. Bangar, B. Draper, R. Beveridge, and J. Ruiz. Exploring the use of gesture in collaborative tasks. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 2990–2997, New York, NY, USA, 2017. ACM.

[25] I. Kokkinos, R. Alp Güler, N. Neverova. Densepose: Dense human pose estimation in the wild. *arXiv*, 2018.