Chapter 3

Full Hand Keypoints Detection

3.1 Overview

The human hands can be considered one of the important parts of the human body as it allows them to be a tangible source for interaction and manipulation of objects in the surroundings. The dexterous nature of the human hand not only allows it to manipulate objects but also as a source of information by forming a gesture that has a certain predefined meaning. The high degree of freedom (DOF) [117] of the human hand allows for intuitive and natural interaction. Hand gestures are being already used for human-to-human interaction. Given the capabilities of a human hand, computer vision researchers have tried to study the hand and its joint motions. Various methods were proposed to infer the types of hand action and device ways to enable it to be used for interaction with computers and robots. To utilize the full capabilities of human hands for HCI requires the development of a system hand that can capture hand motion and interprets all the conveyed meaning.

The information conveyed by a hand is through poses, which may include extending or folding the fingers, positioning the wrist in a particular orientation, showing or hiding the palms, and some other motions. An interpretation system should be able to detect the positions of all relevant hand joints and guess the meaning of a particular hand

pose. This complete flow of interpretation of hand motion is referred to as hand pose estimation.

The vision-based hand pose estimation is a process of finding the position of a few predefined hand joints (also called keypoints) in a still image or video frame—the position and orientation of these keypoints in the 3D space form a hand pose. Usually, a set of 21 predefined hand joints is considered for its pose estimation. However, no guideline specifies how many keypoints are required to estimate the hand pose. The choice of this value is at the discretion of the researchers and also can be application-dependent. The number of keypoints can be as low as five or as high as twenty-five. In our study, we have considered the number of keypoints that defined hand pose to get twenty-one. The twenty-one keypoints include five fingertips, the centers of 15 finger joints, and the base of the palm (wrist point). The hand image shown in Figure 2.4 provides the position of these twenty-one keypoints. However, the vision-based markerless hand pose estimation is challenging, especially from the perspective of the RGB image. Appearance ambiguities, self-occlusions, and occlusion due to objects in the image are some factors that must be dealt with for an accurate hand pose estimation.

An increase in popularity and success of DNN-based approaches, in various CV fields, has drawn researchers' attention to exploring its usage in hand pose estimation [10, 13, 14, 20, 84, 118–122]. The traditional strategy to estimate the hand pose is to localize the hand region of interest (ROIs) in the input image, which is then followed by an independent pose estimation step. Using CNN-based object detection methods the ROIs are localized using bounding boxes and then perform hand pose estimation on the cropped image. However, the aforementioned approach requires more computational resources and slows down the process of hand pose estimation. Furthermore, the aforesaid two-step approach requires considerable when it is essential to estimate the hand pose of more than one hand. Besides this, a two-stage strategy is unsuitable

for applications requiring the interactive-hands. Moreover, hand keypoints detection accuracy is dependent on the robustness of a hand localization process.

In this chapter, we describe two CNN-based methodologies of complete twenty-one hand keypoints detection from a monocular RGB image. In the next section, brief details of work conducted in this area are provided.

3.2 Related Works

The human hand can act as a natural and non-contact input source in many human-computer interaction (HCI) applications. This feature of hand is of great interest in the field of computer vision. Several studies have been conducted to solve hand pose estimation, and various solutions have been proposed [6, 7]. However, hand pose estimation from a color image is challenging due to occlusion, the similarity of fingers, unconditional lighting, background, etc. The vision-based interactions are natural and intuitive. With the introduction of portable depth sensors, research in the field of vision-based hand pose estimation has picked up the pace [123–126]. However, a depth sensor is effective only in an indoor environment, and the depth-maps are noisy [12, 14, 20]. Thus, the scope of a depth sensor is limited. RGB images present an alternative to depth-map for hand pose estimation.

As compared to the depth-map-based methods very less number of methods use RGB images for hand pose estimation. Athitsos and Sclaroff [127] proposed a method based on edge maps and chamfer matching for pose estimation. The use of egocentric vision methods for hand pose estimation has been proposed by Rogez *et al.* [128] and Baydoun *et al.* [129]. The method in [128] used a hierarchical regression technique whereas, a graphical approach is followed in [129]. Grzejszczak *et al.* [16] has used distance transform and templates for hand keypoints detection from an RGB image.

The research on vision-based pose estimation (both, human and hand) has been boosted by a CNN-based architecture's powerful learning ability. Several CNN models have been proposed so far to estimate 2D human poses such as SpatialNet [3], Convo-

lutional Pose Machines (CPM) [83], Stacked hourglass networks [130], and others have been adopted for hand pose estimation. A very popular method that uses CNN to estimate hand pose from RGB images has been proposed by Zimmermann *et al.* [20]. The authors proposed 3D hand pose estimation using 2D hand keypoints detection. They performed heatmaps regression to obtain the 2D hand keypoints. Their model was trained on synthetic hand images and performed well on a synthetic dataset. But, similar performance is not observed on real-world images due to the domain gap between the two. Simon *et al.* [84] also proposed a method for hand keypoints detection using heatmap regression. The authors followed the Convolutional Pose Machine (CPM) [83] architecture in their network design. The detector is not robust in the detection of hand keypoints. The methods proposed by Zimmermann *et al.* [20] and Simon *et al.* [84] are designed for hand pose estimation from cropped hand image and hence dependent on the accuracy of hand localization mechanism.

Similar to these methods, Li *et al.* [85] and Guo *et al.* [118] proposed methods for 2D hand pose estimation for monocular RGB images using cropped hand images. Guo *et al.* [118] use HR-Net [131] to perform heatmap regression and Li *et al.* [85] adopted latent heatmap regression technique proposed by Iqbal *et al.* [132] to estimate the 2D hand pose. Recently, a few methods that perform 2D hand pose directly from full-size images have been proposed. Wang *et al.* [13] proposed a two-stage cascaded CNN architecture for hand pose from a single RGB image. They performed hand mask prediction in the first stage. The second stage is based on CPM architecture which performs heatmap regression on the features map generated in the first stage. Another encoder-decoder-based architecture called SRHandNet was proposed by Wang *et al.* [14] to estimate 2D hand pose from full-size color images. This method is also based on heatmap regression. A different approach is adapted by Li *et al.* [133] to estimate 2D hand pose from the RGB images. They use the object keypoints similarity (OKS) technique using hand templates (called pose anchors) to detect hand keypoints. The

use of pose anchors limits the use of this technique only to a few hand postures.

Another approach, known as holistic regression [134], aims to estimate pose without the need to generate any intermediate representations (pixel-wise classification). However, holistic regression is unable to generalize and translation variance diminishes the predicted results [135]. In our previous work, we have designed a grid-based method for the detection of partial hand keypoints [122]. It deals with the detection of fingertips only and it can be extended to detect all 21 hand keypoints. But it will increase the computational cost of the DNN model. Therefore, in this work, we are proposing a new method for 2D hand pose estimation from full-size images using a single-stage CNN model. The proposed work does not require hand localization except in the case of small hands. Additionally, the proposed method is free of any hand template or anchors and therefore, it is effective for various hand postures.

3.3 Single Hand Keypoints Detection

The objective followed in this section is to locate the hand joints (also called keypoints) from a full-size monocular RGB image. The process makes use of a single-stage CNN architecture to estimate the 2D hand pose. The framework of the complete process is shown in Figure 3.1. In the first feed-forward, shown as a blue dash line, a full-sized image is used as input to the CNN model. The output of CNN is decoded to obtain the coordinates of the hand-bounding box and hand keypoints. A single forward pass is sufficient for hand pose estimation. However, the hand pose estimation accuracy decreases if the hand size is small. The direct detection of hand pose for small hands is challenging. To deal with such cases, we use bounding box coordinates obtained from the first feed-forward to crop the hand region from the full-size image. The cropped hand image is then scaled to fit the network's input size. In the second forward pass through the same CNN network, shown as a red dashed line in Figure 3.1, estimated hand pose accuracy for small hands is improved. The details of the hand keypoints detection and feedback inference process are provided in the following sub-sections.

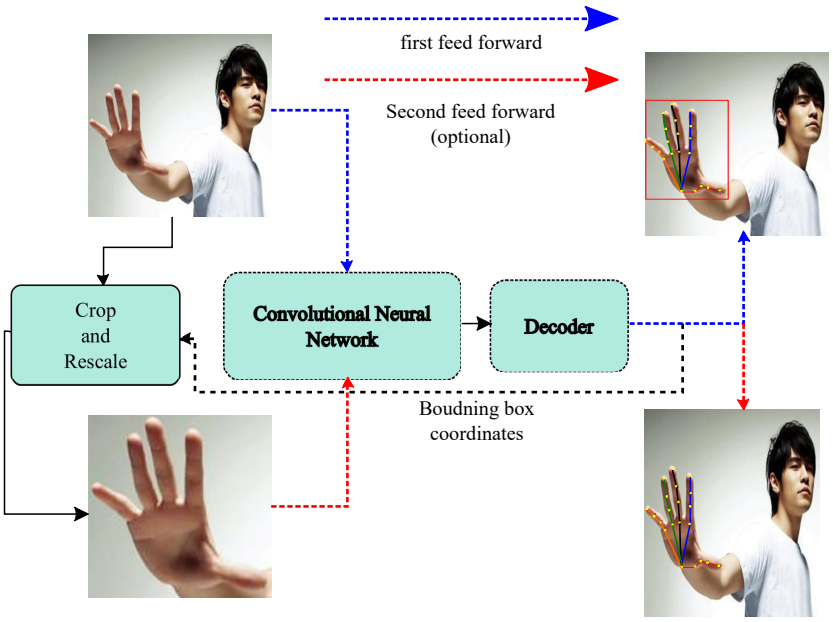


Figure 3.1: Framework for single hand keypoints detection.

3.3.1 Working principle

In a given color image $I \in \mathbb{R}^{w \times h \times 3}$, the 2D hand pose is defined by the position of a set of K hand joints in 2D space. The 2D pixel coordinate of a keypoint is represented by point $p_n = (x_n, y_n) \in \mathbb{R}^2$ where $n \in [1, K]$. The value of $K = 21$. In order to predict the coordinate of p_n for $n \in [1, K]$, we divide the resized image $I' \in \mathbb{R}^{s \times s \times 3}$ into virtual non-overlapping patches of equal size. Each of these grid cells represents a probability indicating the presence of keypoint p_n. If the probability of a grid cell is above the threshold value, δ_p, it indicates the presence of at least one keypoint in that patch of the image. For example, as illustrated in Figure 3.2, a 16×16 grid divides the image into small patches of equal dimension. The white cells in the rightmost image of Figure 3.2 represent cells having high confidence in enclosing at least one keypoint. The grid cell having no hand keypoint is shown in black. The final position of a keypoint is calculated by estimating its offset value relative to the corresponding grid cell.

Suppose the image I' has been divided by the grid G into $M \times M$ patches. The position of the n^{th} keypoint, $p_n = (x_n, y_n)$, is given as

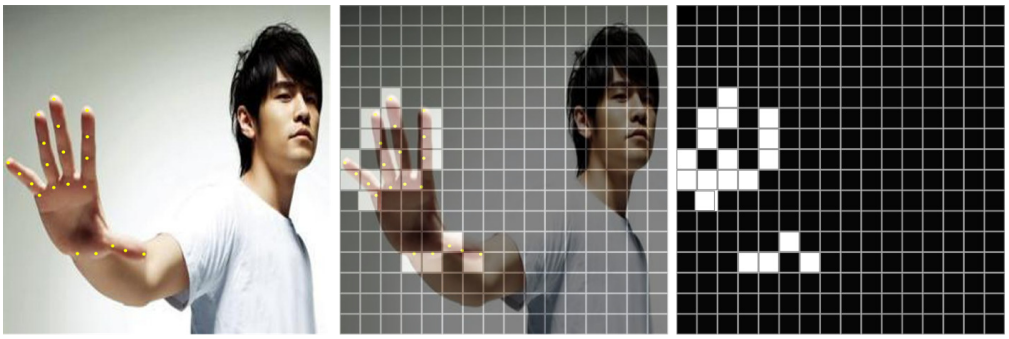


Figure 3.2: Illustration of the process of single hand keypoints detection.

$$x_n = \frac{s}{M}(i + X_{kn}), \tag{3.1}$$

$$y_n = \frac{s}{M}(j + Y_{kn}) \tag{3.2}$$

where, $i \in [0, M-1]$ and $j \in [0, M-1]$ are the indexes for $G_{i,j}$ grid cell in which the point p_n lies. $X_{kn} \in [0, 1]$ and $Y_{kn} \in [0, 1]$ are the normalized offset values for the n^{th} keypoint relative to $G_{i,j}$ grid cell along the width and height of the image, respectively. The offset values are normalized by the edge length of a grid cell.

There is always a chance that more than one hand keypoint can lie in the same image patch. To deal with such cases, each hand keypoint has been assigned a unique label (which is the order in which they are defined in a dataset). We then perform multi-label classification to identify all hand keypoints present in a given grid cell $G_{i,j}$.

In total, three different operations are being performed to locate the positions of visible hand joints in a given color image. Those are

1. Computing the probability of keypoints present in a grid cell,
2. Performing multi-label classification to differentiate between hand joints, if more than one lies in the same grid cell, and
3. Computing the offset values (X_{kn}, Y_{kn}) of a keypoint position relative to the bound of the corresponding grid cell.

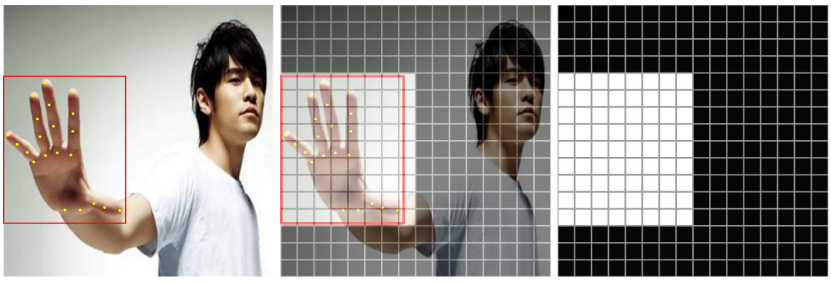


Figure 3.3: Illustration of the process of single hand bounding box detection.

3.3.2 Hand localization

The same grid structure has been previously used for keypoints detection for hand localization. To detect the hand bounding box coordinates of the hand in the image, we predict the confidence score, $C \in [0, 1]$, of each grid cell covering the hand region. Here, the hand region in terms of grid cells is defined as the group of grid cells covered by the bounding box of the hand assuming the grid structure is placed on the image. For example, as shown in Figure 3.3, the white cells represent the group of grid cells covering the hand region. We use a threshold value, $\delta_c \in [0, 1]$, to segregate the high and the low confidence grid cells and compute the top and bottom corner coordinates of the hand-bounding box.

For a input RGB image of size $s \times s$, the top-corner coordinates, (x_{top}, y_{top}) , of the bounding box can be calculated as

$$x_{top} = \frac{s}{M}u, \quad y_{top} = \frac{s}{M}v \tag{3.3}$$

where,

$$u = \Big\{ j, \text{for } \mathbf{C} \geq \delta_c \; AND \; \min_{\forall i,j} i + j,$$

and

$$v = \Big\{ i, \text{for } \mathbf{C} \geq \delta_c \; AND \; \min_{\forall i,j} i + j.$$

Similarly, the bottom-corner coordinates, (x_{bot}, y_{bot}), of the bounding box can be calculated as

$$x_{bot} = \frac{s}{M}u', \quad y_{bot} = \frac{s}{M}v' \tag{3.4}$$

where,

$$u' = \Big\{ j, \text{for } \mathbf{C} \geq \delta_c \; AND \; \max_{\forall i,j} i + j,$$

and

$$v' = \Big\{ i, \text{for } \mathbf{C} \geq \delta_c \; AND \; \max_{\forall i,j} i + j.$$

3.3.3 Feedback Inference

Direct estimation of 2D hand pose is difficult when a hand occupies just a fraction of the image. Therefore, to facilitate accurate pose estimation for small hands, we use bounding box coordinates of the hand obtained after the first forward pass through the proposed network. Then the hand region is cropped and scaled to fit the network's input size. Next, the positions of hand keypoints are estimated using the same network in the second forward pass. This process is referred to as *feedback inference*. However, it should be noted that the feedback inference is only performed if the detected hand is small. A hand is considered to be small, if $\lambda_h \leq 0.3$, and $\lambda_w \leq 0.3$. Here, λ_h is the ratio of bounding box height to image height and λ_w is the ratio of bounding box width to image width. The feedback inference improves the accuracy of hand pose estimation.

3.3.4 Network architecture

A CNN model is trained to learn features from the entire image and simultaneously perform both the processes of hand pose estimation and hand localization. The ar-

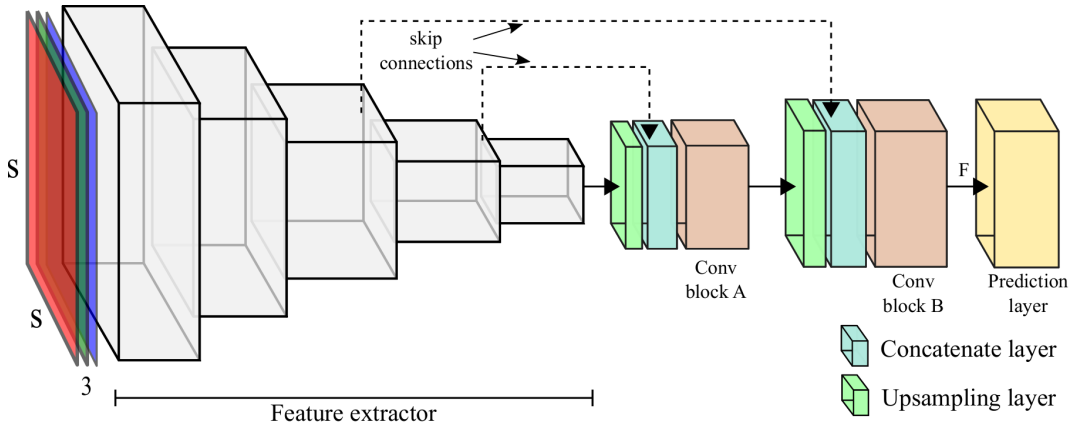


Figure 3.4: The CNN architecture used for single hand keypoints detection.

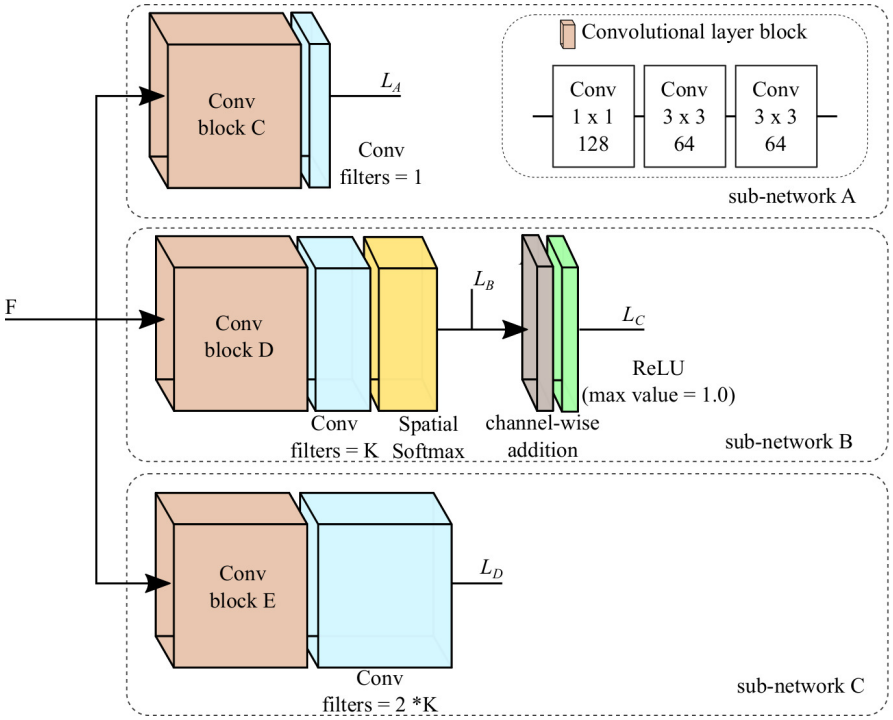


Figure 3.5: The CNN architecture used for single hand keypoints detection.

chitecture of the CNN model used to estimate the position of hand joints is shown in Figure 3.4. The architecture of the proposed model is fully convolutional [136].

The model mainly consists of a feature extractor and predictor. The feature extractor is the backbone complete architecture. We have tried various different DenseNet architectures namely DenseNet121, DenseNet169, and DenseNet201 as the backbone.

The backbone model reduces the spatial resolution of the input image by a factor of 32. Hence, in order to increase the spatial resolution of the output feature map, we used two sequentially connected upsampling blocks (it consists of an upsampling layer followed by a convolutional block as shown in Figure 3.4). These upsampling blocks increase the resolution of feature maps obtained from the backbone layer by a factor of 4. A prediction layer is then connected to the last upsampling block, which computes different probability scores required to obtain bounding box coordinates and hand keypoints position. The configuration of the prediction layer is shown in Figure 3.5. The prediction layer consists of three parallel-connected sub-networks: sub-network A, sub-network B, and sub-network C. All of them are fully convolutional. The *sub-network A* computes the probabilities of multiple grid cells enclosing the hand region. The other two sub-networks compute the probability and offsets required to get the hand keypoints' position. At the time of inference, a decoder extracts the hand-bounding box and keypoints coordinates using the multiple scores provided by the prediction layer.

3.3.5 Decoder

The proposed architecture produces different probabilities for hand localization and keypoints position detection. The output of sub-network A is used to obtain the hand-bounding box coordinates. The details of obtaining bounding box coordinates from the output of the sub-network are provided in Section 3.3.2. Next, the outputs of sub-network B and sub-network C are decoded as explained in Section 3.3.1 and are combined to obtain hand keypoints coordinates.

3.3.6 Model optimization

To train the proposed model, we compute four different losses L_A, L_B, L_C, and L_D. The loss value L_A is then computed for the case hand localization. The losses L_B and L_C are part of sub-network B. The loss L_D is the regression loss value used to optimize the parameters of sub-network C. These losses are represented in Figure 3.5. The total loss is given as

$$L_{total} = L_A + L_B + L_C + L_D \tag{3.5}$$

$$L_{MSL} = \sum \hat{M}.(\hat{y} - y)^2 + \sum (1 - \hat{M}).((\hat{y} - y)^2, \tag{3.6}$$

where

$$\hat{M} = \frac{\hat{y}}{\hat{y} + \epsilon}, \tag{3.7}$$

and, $\hat{y}$ and y is the ground and predicted feature maps, respectively. $\epsilon = 10^{-7}$ is used in (3.7) to avoid division by zero.

The L_A, L_B, and L_C are computed using masked squared loss function L_{MSL} defined in (3.6) while the L_2 loss function is used in L_D. To compute the losses the ground-truth 2D hand keypoints position are first encoded into four different output formats as produced by the proposed model. Each encoded ground truth has the same shape as the network's output.

The network parameters were optimized with Adam optimizer [137]. The value for learning rate is 2×10^{-4}. The learning rate scheduler strategy proposed in [108] is used during model training. The model was trained for 300 epochs with a batch size of 8. The Tensorflow [138] framework was used to implement the CNN architecture. The experiments were performed on a single Nvidia GTX 1080 GPU.

3.3.7 Experimental Analysis

Performance evaluation metric. The PCK metric defined in Section 2.5.4 is used for performance evaluation of the hand keypoints detection algorithm. Additionally, the Intersection over Union (IOU) metric is used to quantify the performance of the

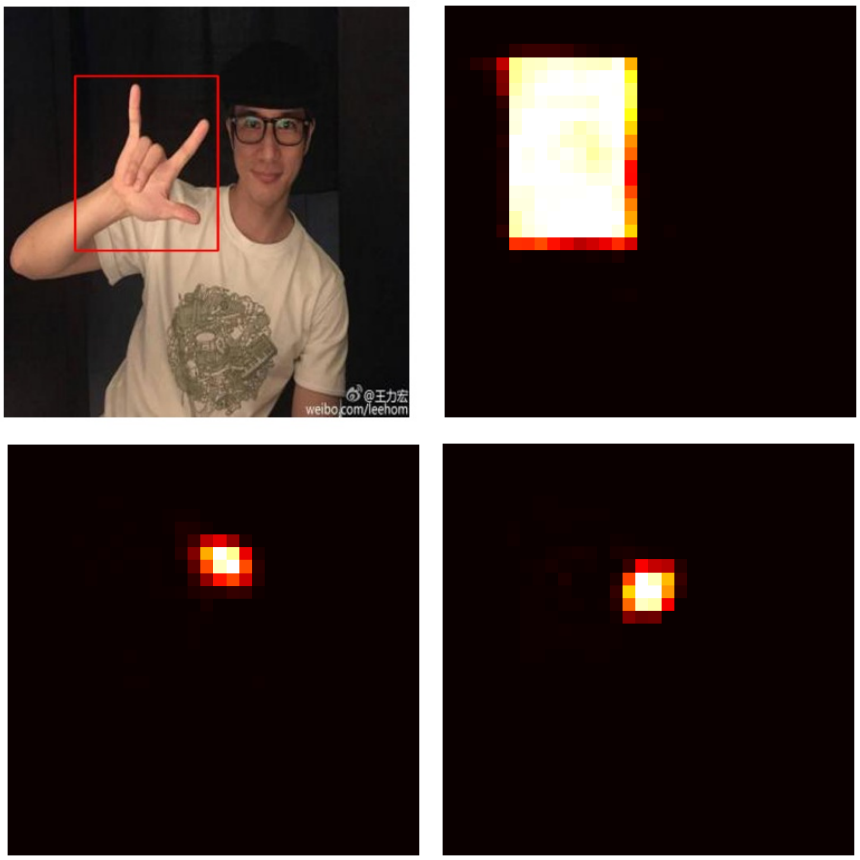


Figure 3.6: The figure demonstrates heatmaps obtained to detect various hand features. The top left image shows the hand ROI and the model's ROI detection probability is shown using the heatmap in the top right. The bottom two images show the two hand keypoints detection probability in terms of the heatmap.

hand-bounding box generation methodology.

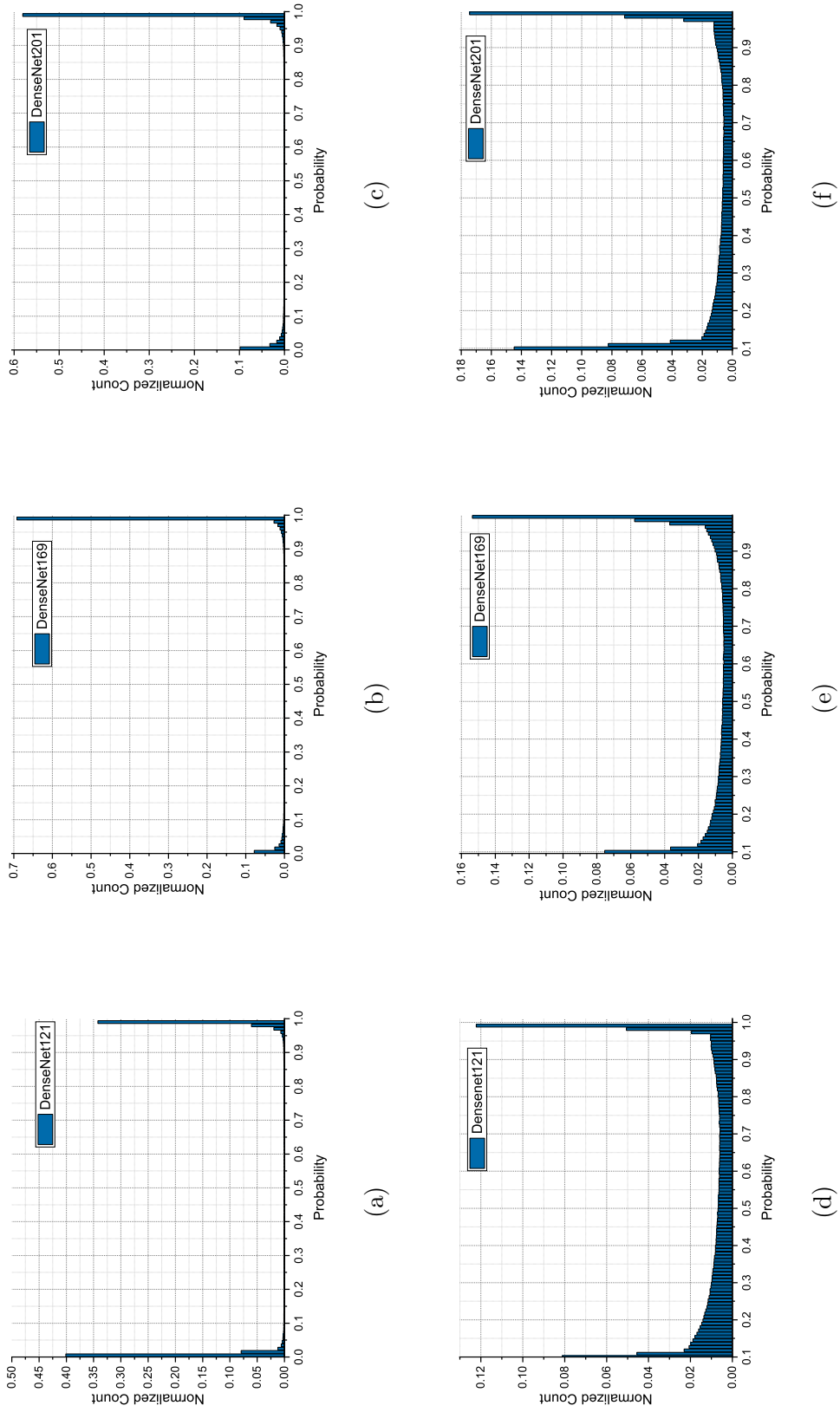


Figure 3.7: The normalized histograms obtained from probability map generated by the CNN model.

3.3.7.1 Threshold estimation

The DNN model output is probability maps. The probability maps are of two different types, one is for hand ROI detection, and another is for all hand keypoints detection. The heatmap representation of the probability map obtained on a sample image for hand ROI and two keypoints are shown in Figure 3.6.

Two different threshold values are used during the decoding process to calculate the hand bounding box coordinates and the keypoints' spatial position from the hand ROI and keypoints' probability maps, respectively. To calculate these threshold values, we used the probability maps for all the test samples and plotted the histogram. The normalized histogram obtained from hand ROI probability maps by using different backbone models on test samples is shown in Figure 3.7ato 3.7c. Similarly, the Figure 3.7d to 3.7f shows the normalized histogram obtained for all 21 hand keypoints on test samples. Using these two histogram plots and performance metrics, we estimated the threshold value for hand ROI detection to be 0.8 and the threshold values for hand keypoints position estimation to be 0.70.

3.3.7.2 Feedback inference

The feedback inference is effective in cases where it is difficult to obtain acceptable hand keypoints detection results from a full-size RGB image. With the use of the

(a) (b)

Figure 3.8: Example showing improvement in keypoints detection accuracy with feedback inference

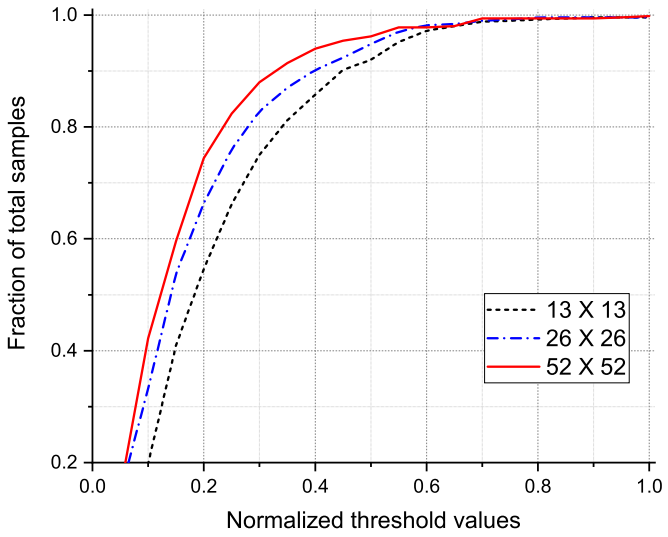


Figure 3.9: The PCK curve for different grid sizes.

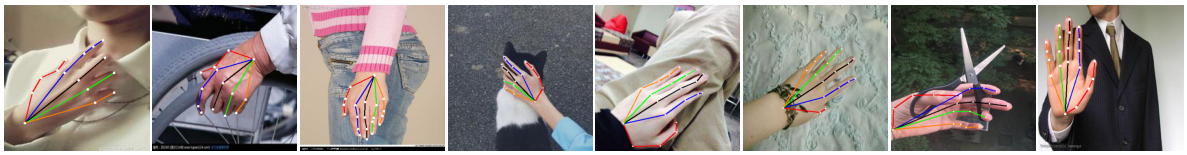
feedback inference technique, the process of hand keypoints detection transforms into a two-step process. The problem of undetected or incorrectly located keypoints is rectified by using this technique. The effectiveness of feedback inference is studied on the OneHand10K dataset [13] by categorizing images in the dataset into two groups (normal and small) based on the hand size. The process of defining a hand as small is discussed in Section 3.3.3. The examples shown in Figure 3.8 illustrate the improvement in keypoints detection observed using feedback inference on sample test images. Since both hand- localization and keypoints detection procedures have been implemented using a single CNN model, it overcomes the burden of training a separate network for hand localization. Additionally, it conserves computational resources and improves run-time efficiency.

3.3.7.3 Effect of the grid size

We experimented with grids of different sizes viz. 13×13, 26×26, and 52×52. The PCK curves for each grid size on the OneHand10K dataset [13] are shown in Figure 3.9. It is clear from this figure that the accuracy of keypoints detection increases as the grid

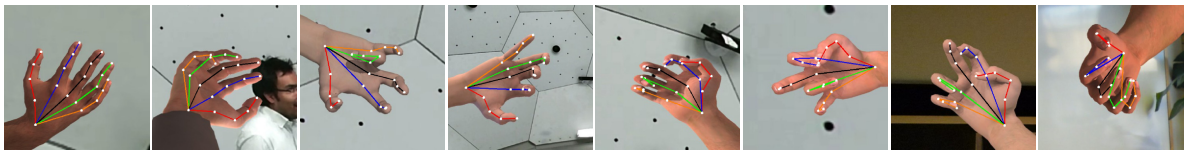
size increases. The reason is that the larger the grid size, the smaller the grid cell size; thus, the grid cell containing will be close to the actual keypoints position. Additionally, the error in offset calculation arising due to regression will be minimized with a larger grid size. The same can be illustrated with an example shown in Figure 3.10. The top row shows the image with a grid size of 13×13, the middle row has a grid size of 26×26, and the bottom row has a grid size of 52×52. The bounding box shown in red becomes closer to the hand as the grid size is increased. However, the size of the grid has correspondence with the input image size. Therefore, the input image size must be of high resolution to obtain a larger output grid size. As we know, the convolutional layer performs spatial operations wherein filters are moved over the image. Thus, its computation time is dependent on the input image size. Therefore, we choose the grid size to be 52×52 in the proposed work to have better accuracy and computational time. The input image size dimension for the aforementioned case is 416×416.

Figure 3.10: The effect of grid size of hand keypoints detection process.

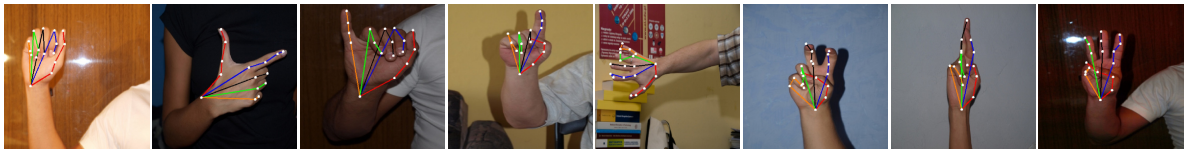


(a) OneHand10K [13]

(b) STB [21]



(c) CMU [139]



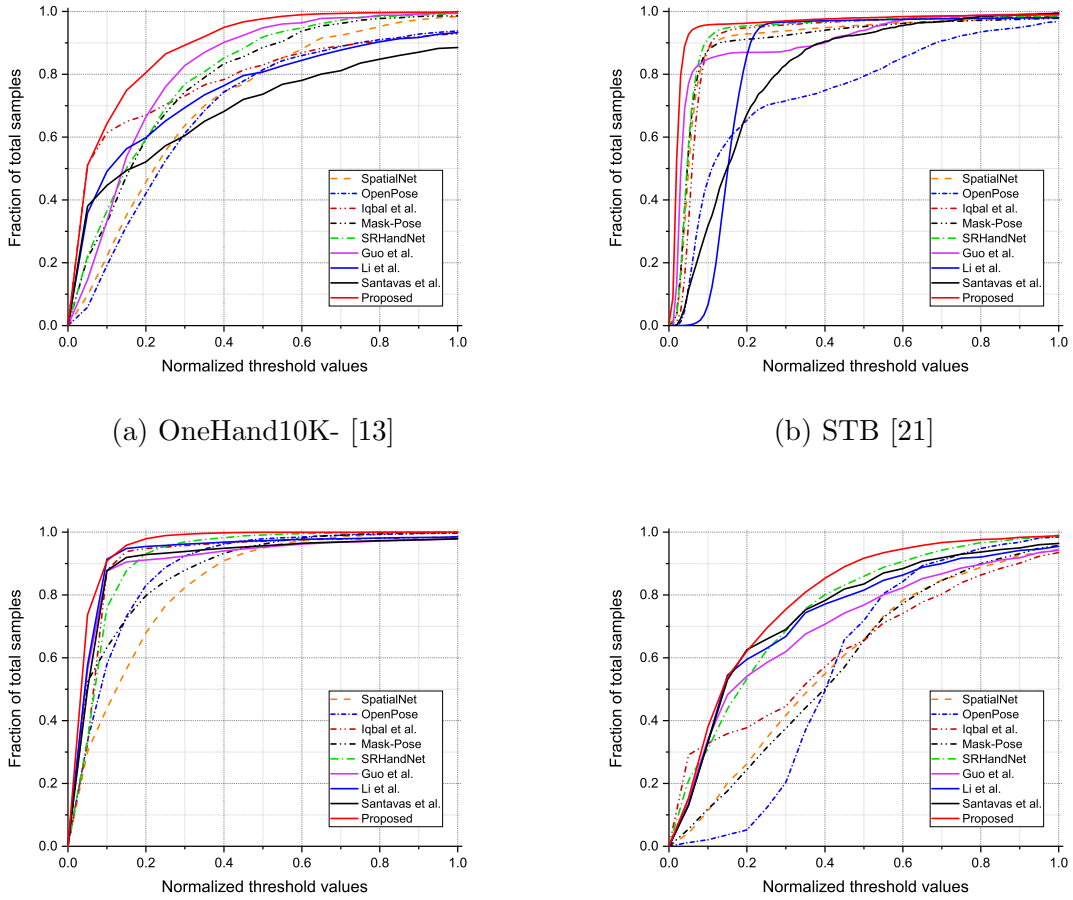
(d) HGR [16]

Figure 3.11: Sample results with grid-based hand keypoints detection algorithm on three different datasets.

3.3.7.4 Comparison

We have compared the results of the proposed method with several state-of-the-art methods available for hand keypoint detection on RGB images. The recent methods for hand pose estimation namely OpenPose [84], Mask-Pose [13], and SRHandNet [14] were selected. Additionally, methods proposed by Iqbal *et al.* [132], Guo *et al.* [118], Li *et al.* [85], and Santavas *et al.* [134] are also used for the detection of hand keypoints.

All these models were trained on the same datasets to have a fair comparison. As the methods in [3, 13, 14, 84, 132] are based on heat-map regression; at inference, we set the threshold value to 0.2 and obtained joints location from the heat maps. A few sample results obtained with the proposed method on the test set of the four different



(a) OneHand10K- [13]

(b) STB [21]

(c) CMU [139]

(d) HGR [16]

Figure 3.12: Comparison of proposed grid-based keypoints detection method of various datasets

datasets are presented in Figure 3.11. Each finger is represented with a different color. We used red, blue, black, green, and orange colors to represent thumb, index-, middle-, ring-, and pink- finger, respectively. The finger joint locations are marked with a white dot.

The quantitative comparison is conducted using (2.12). The PCK curves for each method on the OneHand10K dataset are shown in Figure 3.12a. The solid red curve in Figure 3.12a quantifies the performance of the proposed method. We have furnished the details of the mean averaged PCK in Table 3.1. The mean averaged PCK, at $\sigma = 0.2$,

Table 3.1: **Peformance comparison of the grid-based hand keypoints algorithm with different methods. The PCK value is calculated for the threshold value of $\sigma = 0.2$**

Method	OneHand10K [13]		STB [21]		CMU [139]		HGR [16]	
	PCK	AUC	PCK	AUC	PCK	AUC	PCK	AUC
SRHandNet [14]	0.6132	0.7787	0.9346	0.9186	0.9310	0.8932	0.5331	0.7410
Li *et al.* [85]	0.5986	0.7312	0.8602	0.8225	0.9546	0.9049	0.5951	0.7208
OpenPose [84]	0.4208	0.6734	0.6542	0.7588	08303	0.8602	0.1519	0.5672
Mask-Pose [13]	0.5940	0.7620	0.9118	0.8975	0.7984	0.8607	0.2442	0.5781
Santavas [134]	0.5215	0.6740	0.6725	0.8004	0.9288	0.8865	0.6256	0.7345
Guo *et al.* [118]	0.6640	0.7976	0.8698	0.8994	0.9118	0.8845	0.5410	0.6878
SpatialNet [3]	0.4560	0.7003	0.8287	0.9002	0.6788	0.8148	0.2625	0.5870
Iqbal *et al.* [132]	0.6698	0.7649	0.9282	0.9046	0.9482	0.8892	0.3774	0.6174
Grid-Based(Proposed)	**0.7687**	**0.8478**	**0.9631**	**0.9416**	**0.9791**	**0.9313**	**0.6312**	**0.7764**

is higher than the other methods. All results, including the qualitative analysis, and the PCK, demonstrate the superiority of the proposed method.

Although the HGR dataset provides ground-truth annotations for 25 hand keypoints, we used only 21 hand keypoints, and the rest of the 4 keypoints, marked as 'ConcavePoint', are ignored during the comparison. For testing on CMU hand dataset [139], we considered only two subsets marked as '*synth2*' and '*synth3*' because '*synth2*' and '*synth3*' comprises of 3,243 and 2,348 synthetically generated single hand images, respectively.

The PCK curves on CMU hand dataset [139] and HGR dataset [16] are shown in Figure 3.12c and Figure 3.12d, respectively. The PCK at $\sigma = 0.2$ and AUC for these two datasets is presented in Table 3.1. It can be deduced from all these results that the performance of the proposed method, on different types of datasets, is superior to the existing methods.

3.3.8 Discussion

The proposed grid-based algorithm is an acceptable solution for 2D hand pose estimation from the monocular RGB image. The results demonstrate that hand keypoints can be directly located, and the hand localization step is redundant. The proposed method is effective against the variation in lighting conditions, skin color, subject's

gender or age, background, etc. It can even work with different hand postures and hand sizes. The network's speed is acceptable enough to deploy it in gesture-based applications such as virtual typing, airwriting, etc. However, there are a few limitations of the proposed method. First, similar to methods in [3, 13, 14, 84], the proposed model is trained only for hand keypoints detection from single hand color images. Second, only visible hand joints can be located, and hand joints hidden from view are not detected.

3.4 Double Hand Keypoints Detection

The same objective that is described in Section 3.3 is being carried over in this section. The additional objective followed here is to increase the hand keypoints detection and extend the method for double hand keypoints detection. Instead of trying to estimate a hand keypoint position by finding a single point on feature maps produced by a DNN model, we try to identify N neighborhood points to a possible keypoint. Using the N neighborhood points, we determine the probable location of a keypoint. The idea to follow the neighborhood points approach arises from noticing the manual process of ground-truth annotations. Given a certain number of expert annotators, if each one of them marks keypoint positions independently of each other, then individually marked positions will probably differ from each other. However, they will be in close proximity to each other (can be called neighborhood points). Furthermore, the actual keypoint position will be very close to the centroids of all these points. Based on this hypothesis, we formulate the process of keypoints position estimation as the process of detection of N neighborhood points. Using these neighborhood points, we calculate the probable position of the hand keypoint. Entire details of the proposed method are given below.

3.4.1 Working principle

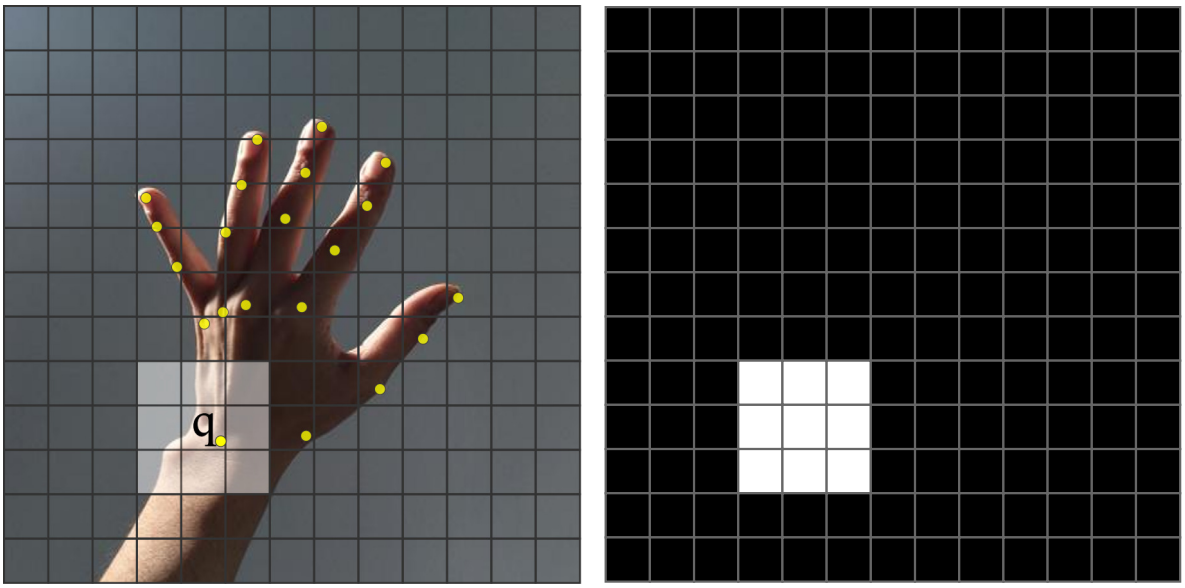
The aim is to estimate hand keypoints directly from a full-size RGB image. The last statement implies that we plan to devise a method that will work without segmenting the hand region from the given monocular RGB image before giving the keypoints position. To meet our objective, we divide the full-size RGB image into $M \times M$ grids.

This grid structure helps in locating hand keypoints in the image. Predictions are made using a CNN-based architecture. The proposed CNN architecture takes a full-size RGB image as input and produces a three-dimensional feature map. The spatial dimension of this feature map corresponds to the aforementioned grid. The information regarding each keypoint position is encoded in separate channels of output feature maps obtained from the CNN model. The design of the proposed CNN architecture enables end-to-end training and faster prediction. The details of the proposed hand keypoints detection are given below.

The input to the proposed CNN model is a monocular RGB image. In a color image $\mathbf{I} \in \mathbb{R}^{w \times h \times 3}$, the 2D hand pose is represented as $\mathbf{p} = \{p_k\}_{k \in K}$. Here, $p_k = (x_k, y_k) \in \mathbb{R}^2$ denotes the 2D pixel coordinates of the *k-th* keypoint in the image. Here, K is the number of keypoints in hand, and $K = 21$ for a single hand is the most commonly used value mentioned in numerous literature.

During the training process of the CNN model, we use the ground-truth keypoints and grid cells to create a pattern. The model learns this pattern, and at inference time, it produces a similar pattern which will be decoded to get the estimated location. As an example, the wrist point marked as 'q' as shown in Figure 3.13a is encoded to the pattern formed by white grid cells shown in Figure 3.13b. We use the center of each grid cell as a reference point to identify the grid cell where a ground-truth hand keypoint lies. To do so, we calculate the Euclidean distance of all ground-truth keypoints with all reference points. The grid cell with the smallest Euclidean distance is the cell where a keypoint is present.

Mathematically, this can be formulated as, if the reference points (centers of grid cells) are represented as $\mathbf{q} = \{q_m\}_{m \in M}$, where $q_m = (x_m, y_m) \in \mathbb{R}^2$ is the 2D-pixel coordinate of the *m-th* point in the image. And, if $d(p_k, q_m)$ represent Euclidean distance between *k-th* keypoint and the *m-th* reference point. Then, the value of m given the value of k for which we get minimum $d(p_k, q_m)$, is used to identify the grid cell where



(a) (b)

Figure 3.13: An illustration of the method used for the estimation of 2D hand keypoints position from an RGB image

the *k-th* keypoint is present.

As stated above, the position of the *k-th* keypoints is estimated with the help of **N** neighbors. After identifying the gird cell where the *k-th* keypoint is present, we use all eight adjacent cells (if the center grid cell where the keypoint lies is not present near the edge) to form neighbors. During the encoding process, all 9 cells (8 neighbors and center cells) are assigned the value of one, and the values of the remaining cells are made zero. The white cells in Figure 3.13b represent the center and eight neighboring cells for the wrist point of the hand shown in Figure 3.13a. This pattern is formed for all K keypoints. We create an array of shapes $M \times M \times K$ during the encoding process.

At inference, we get a feature map of shape $M \times M \times K$ (same as an encoded array). Each cell in the output array represents a probabilistic value in the range $[0, 1]$. We use a threshold value of δ_q to find all such grid cells that possibly represent a group of cells where a possibly a keypoints is present. In addition, we add one extra constraint of a minimum of two cells that should have a value above δ_q and these cells should be

adjacent. For all such neighbor grid cells having a value above the threshold δ_q, we calculate their center coordinates. If $G_{i,j}$ represent one such grid cell then for an image of spatial dimension $s \times s$ its center coordinates (x_m, y_m) can be calculated as

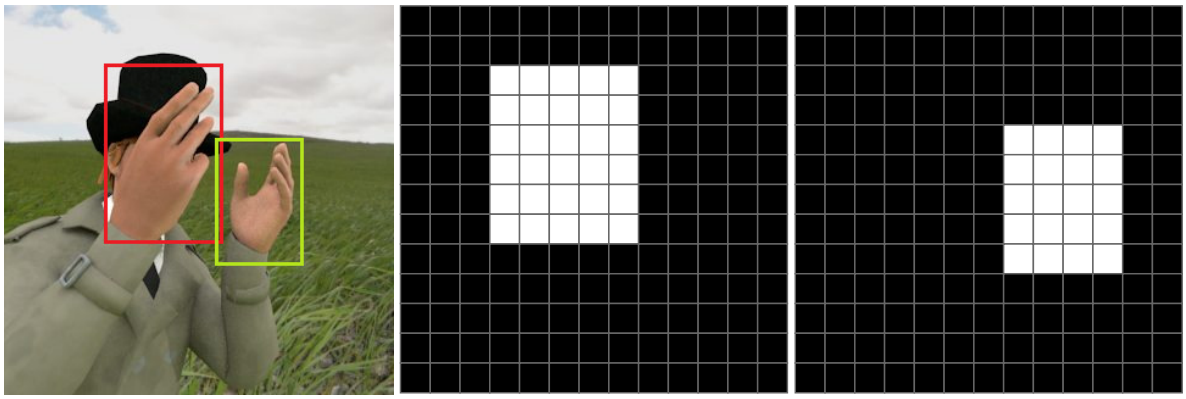
$$x_m = (j + 0.5)\frac{s}{M}, \tag{3.8}$$

$$y_m = (i + 0.5)\frac{s}{M}. \tag{3.9}$$

where, for the grid cell $G_{i,j}$, i and j denote the integer values of the row and column of the grid spanning in the range 0 and $M - 1$. The indexing starts from the top-left corner. The final estimated keypoint coordinate is the centroid of all points obtained during the decoding process.

3.4.2 Hand ROIs Detection

The framework of the proposed hand ROI detection method is similar to the framework of hand keypoints detection. We have designed the whole process for a single user



(a) (b) (c)

Figure 3.14: In order to localize the hand in a color image, the input image is divided into $M \times M$ grid. Then grid cells covering the hand region in the image are identified. Separate channels are used for the detection of each hand. Here, (a) the original image and white cells in the grid represent the hand region for (b) the right hand, and (c) the left hand.

using both of his hands to make gestures. Hence, a maximum of two hand ROIs can be detected with the proposed method. We use $M \times M$ grids as used in hand keypoints detection. During the training process of the CNN model, we encode the ground-truth hand bounding box coordinates to a pattern as shown in Figure 3.14a. Two separate arrays (each for the right and left hand) of shape $M \times M$ are created during encoding for a single user who may be using both of his hands for gesture-based communication. White cells in Figure 3.14b and Figure 3.14c denote groups of grid cells representing the hand ROIs. This group of grid cells will cover the hand region if the grid structure is overlaid on the input image after resizing.

During inference, the model output feature maps which is also an $M \times M$ array. Each cell in the output array represents models' confidence score $\mathbf{C}$ in terms of probability, and hence the value of $\mathbf{C}$ is bounded, i.e., $0 \leq \mathbf{C} \leq 1$. We use a threshold, $\delta_c \in [0, 1]$, to segregate the high and low confidence grid cells. All the grid cells with $\mathbf{C} \geq \delta_c$ are used to calculate the top and bottom corner coordinates of the bounding box.

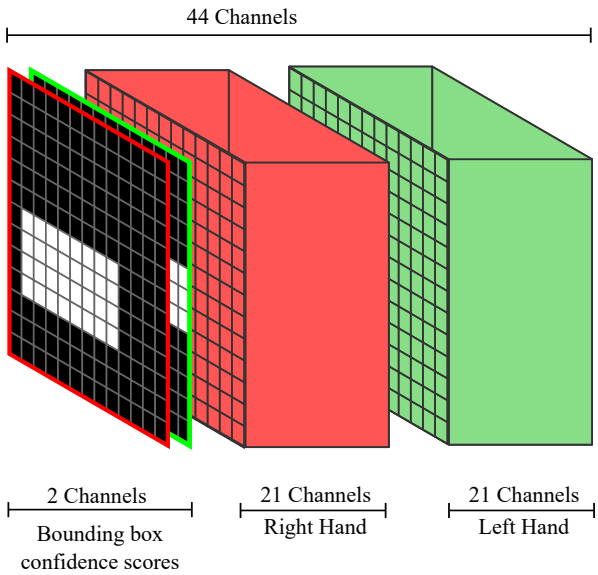


Figure 3.15: The segregated view of the output array of a prediction layer. Separate channels are used for hand localization and keypoints detection of left and right hands. The first two channels are used for Hand ROIs detection and the rest of the others are used for hand keypoints detection.

If $G_{i,j}$ represents a grid cell at index i, j, then for a image of size $s \times s$, the top-corner coordinates, (x_{top}, y_{top}), of the bounding box can be calculated as

$$x_{top} = \frac{s}{M}u, \quad y_{top} = \frac{s}{M}v \tag{3.10}$$

where,

$$u = \Big\{ j, \text{for } \mathbf{C} \geq \delta_c \; AND \; \min_{\forall i,j} i + j,$$

and

$$v = \Big\{ i, \text{for } \mathbf{C} \geq \delta_c \; AND \; \min_{\forall i,j} i + j.$$

Similarly, the bottom-corner coordinate, (x_{bot}, y_{bot}), of the bounding box can be calculated as

$$x_{bot} = \frac{s}{M}u', \quad y_{bot} = \frac{s}{M}v' \tag{3.11}$$

where,

$$u' = \Big\{ j, \text{for } \mathbf{C} \geq \delta_c \; AND \; \max_{\forall i,j} i + j,$$

and

$$v' = \Big\{ i, \text{for } \mathbf{C} \geq \delta_c \; AND \; \max_{\forall i,j} i + j.$$

For each hand, these coordinates are computed from the corresponding grids. All the grid cells for which $\mathbf{C} \leq \delta_c$ are not considered as a hand ROI.

Both the encoded arrays for hand keypoints and ROI detection can be combined to form a single array. Figure 3.15 illustrate the form of an encoded array. If the proposed method is used for a maximum of two hand pose estimation (single user) applications then the shape of the encoded array will be $M \times M \times 2K + 2$. Output feature maps with a similar shape are obtained from the model at inference. The hand keypoints' position and ROIs can be estimated from this feature map using the aforementioned individual decoding processes.

3.4.3 Network architecture

A single-stage CNN network is designed for hand keypoints and ROI detection. The proposed CNN architecture is shown in Figure 3.16. The model is a fully convolutional network [136] and can be trained end-to-end.

The model primarily consists of a feature extractor followed by an up-sampling layer and a prediction layer. We use outputs from three layers of the feature extractor namely $C3,\ C4,$ and $C5$ for predictions. Given, the input image size of $s \times s$, the output strides at $C3,\ C4,$ and $C5$ are $\frac{s}{8},\ \frac{s}{16},$ and $\frac{s}{32}$, respectively. The output feature maps at $C5$ are then fed to an Atrous Spatial Pyramidal Pooling (ASPP) layer [35]. It allows resolution enhancement and enlargement of the field of view of a filter. It also keeps both computation and the number of parameters contained. We added an ASPP layer over the feature extractor layer to increase its receptive field. Followed by the ASPP layer, we use two up-sampling layers which increase the spatial resolution of feature maps by four times. Each up-sampling layer enlarges the features map by two times and the resulting output is added to features maps of the corresponding shape obtained from the feature extractor as shown in Figure 3.16. We use nearest-neighbor interpolation for up-sampling. At the end of the network, we can connect a prediction layer which is a convolutional layer with the number of filters equal to $2K + 2$. The output probabilistic feature map from this layer has the shape as shown in Figure 3.15. The hand-keypoint positions and ROI are estimated by decoding this feature map.

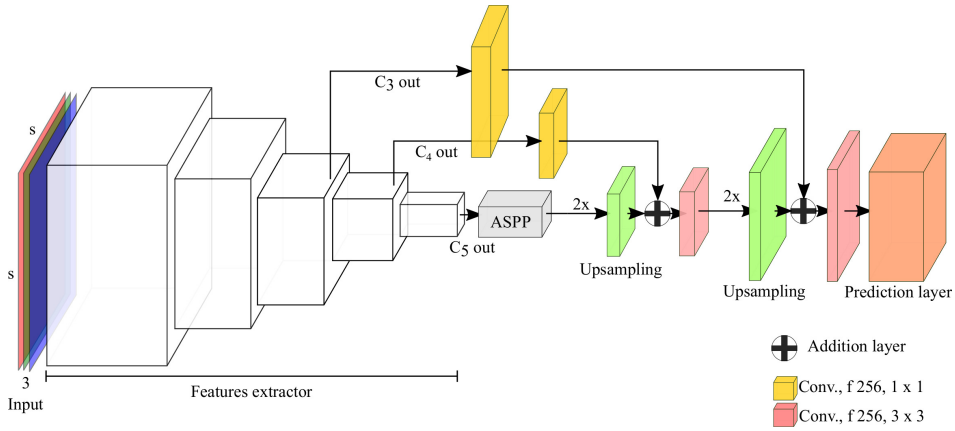


Figure 3.16: The proposed CNN model for simultaneous hand localization and keypoints detection from an RGB image. The details of the network's configuration are provided in Section 3.4.3.

3.4.4 Model optimization

The model that is shown in Figure 3.16 is trained using Nesterov's accelerated Stochastic Gradient Descent (SGD) optimization algorithm. The value of momentum used is 0.9. We applied the learning rate scheduling strategy suggested in [108] with SGD. The value of the initial learning rate is 1×10^{-3} and the power is 0.9. The loss function L_{MSL} used to optimize the model's parameters is defined below

$$L_{MSL} = \sum \hat{M}.(\hat{y} - y)^2 + \sum (1 - \hat{M}).(\hat{y} - y)^2, \tag{3.12}$$

where

$$\hat{M} = \frac{\hat{y}}{\hat{y} + \epsilon}, \tag{3.13}$$

and, $\hat{y}$ and y is the ground and predicted feature maps, respectively. In order to avoid dividing by zero $\epsilon = 10^{-7}$ is used in (3.13) .

During training, the ground-truth keypoint coordinates are encoded to match the output tensors of the corresponding scale as discussed in Section 3.4. Moreover, the

training process includes data augmentation and batch normalization [140]. The batch is chosen to be the maximum value that can fit on the GPU memory. The model is trained for 300 epochs. The proposed CNN model is designed with Keras [141] framework having Tensorflow [138] as the backend.

3.4.5 Experimental Analysis

Performance evaluation metric. The PCK metric defined in Section 2.5.4 is used for performance evaluation of the hand keypoints detection algorithm. Additionally, the Intersection over Union (IOU) metric is used to quantify the performance of the hand-bounding box generation methodology.

The same threshold estimation process described in Section 3.3.7.1 is used here to decode the hand keypoints from the probability mask received from the CNN model.

3.4.5.1 Single vs Multiple neighborhoods

We investigated the effect of the model's performance on keypoints detection accuracy for two cases defined ahead. In case 1, the keypoint positions are estimated from the output feature maps of the proposed model by using just a single point with the highest probability. In case 2, we used nine points (nearest neighbors) to estimate the keypoints' position. We have computed the performance in terms of PCK value at threshold $\sigma = 0.2$ for both of these aforementioned cases and compared them. The obtained values are reported in Table 3.2. The results obtained on different datasets clearly entail that the use of multiple points (nearest-neighbor) to estimate the keypoints' position is better suited than using just a single point. In the cases where just a single point is used for keypoint detection, more divergence is observed between the computed and actual positions.

3.4.6 Comparison

The performance of the method proposed in this section for hand keypoints detection is compared with OpenPose [84], Mask-Pose [13], SRHandNet [14], A2J [126], and SpatialNet [3]. The Mask-Pose architecture is designed to estimate a 2D hand pose

Table 3.2: **The performance of the proposed model in terms of PCK value at** $\sigma = 0.2$

Method	OneHand10K	RHD	InterHand2.6M
Single point	0.6286	0.8765	0.9276
Multiple neighbors	0.6847	0.9010	0.9471

from a single hand. Therefore, we used it for comparison only with those datasets in which a single hand in an image is available. In order to detect keypoints from double hands, the SpatialNet, OpenPose, A2J, and SRHandNet models have been modified. The number of output channels in all these models is changed to 42 so that keypoints from double hands can be detected. The methods in [3, 13, 14, 84] are based on heat-map regression; at inference, we set the threshold value to 0.2 and obtained keypoints position from the heat maps. The A2J [126] model was designed to estimate the 3D hand keypoints position from a depth image. It uses anchor points placed at strides = 4 along the spatial dimension of the image to locate keypoints. The keypoints are located using a 2D CNN model. This architecture has three sub-networks to estimate the 3D position of a hand keypoint. For 2D hand keypoints position estimation, the depth estimation branch from the A2J model is removed. The A2J model is trained with 256×256 image size.

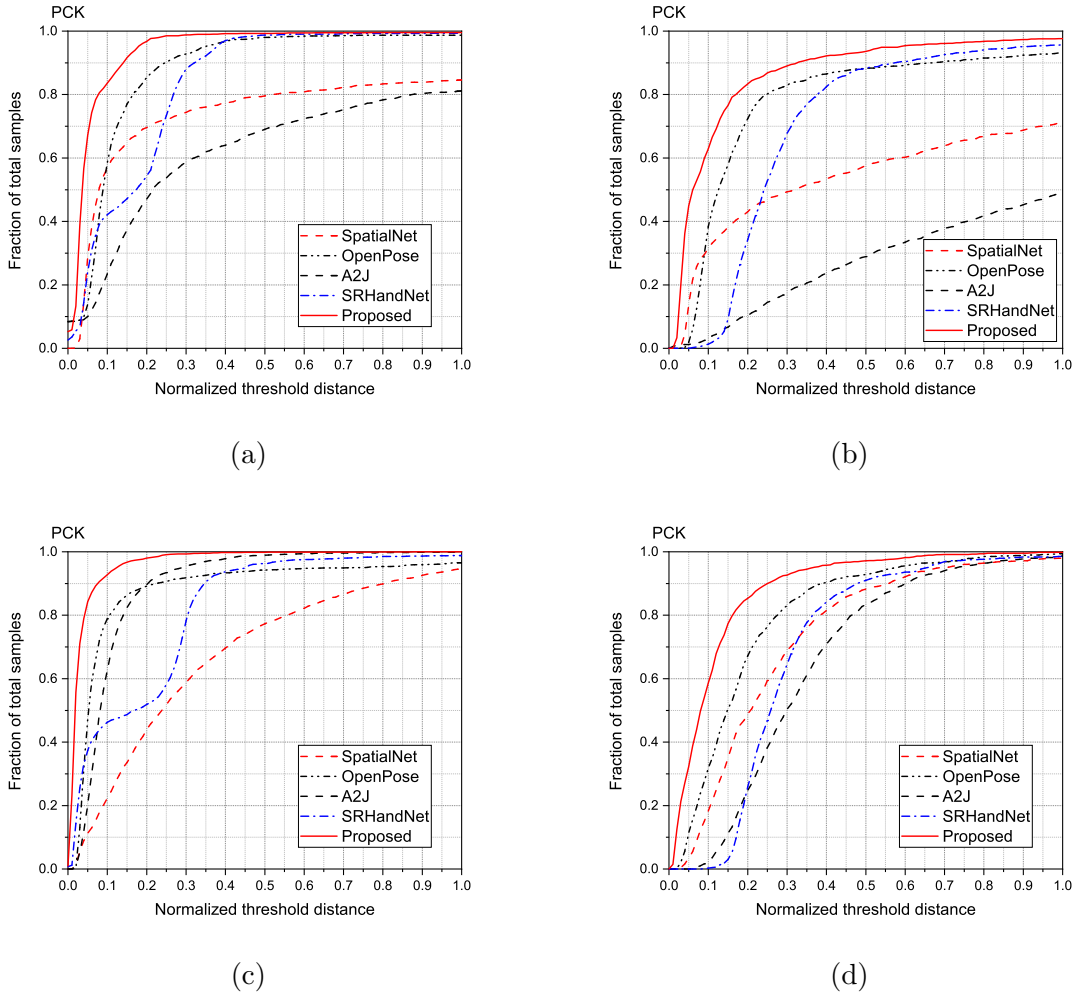
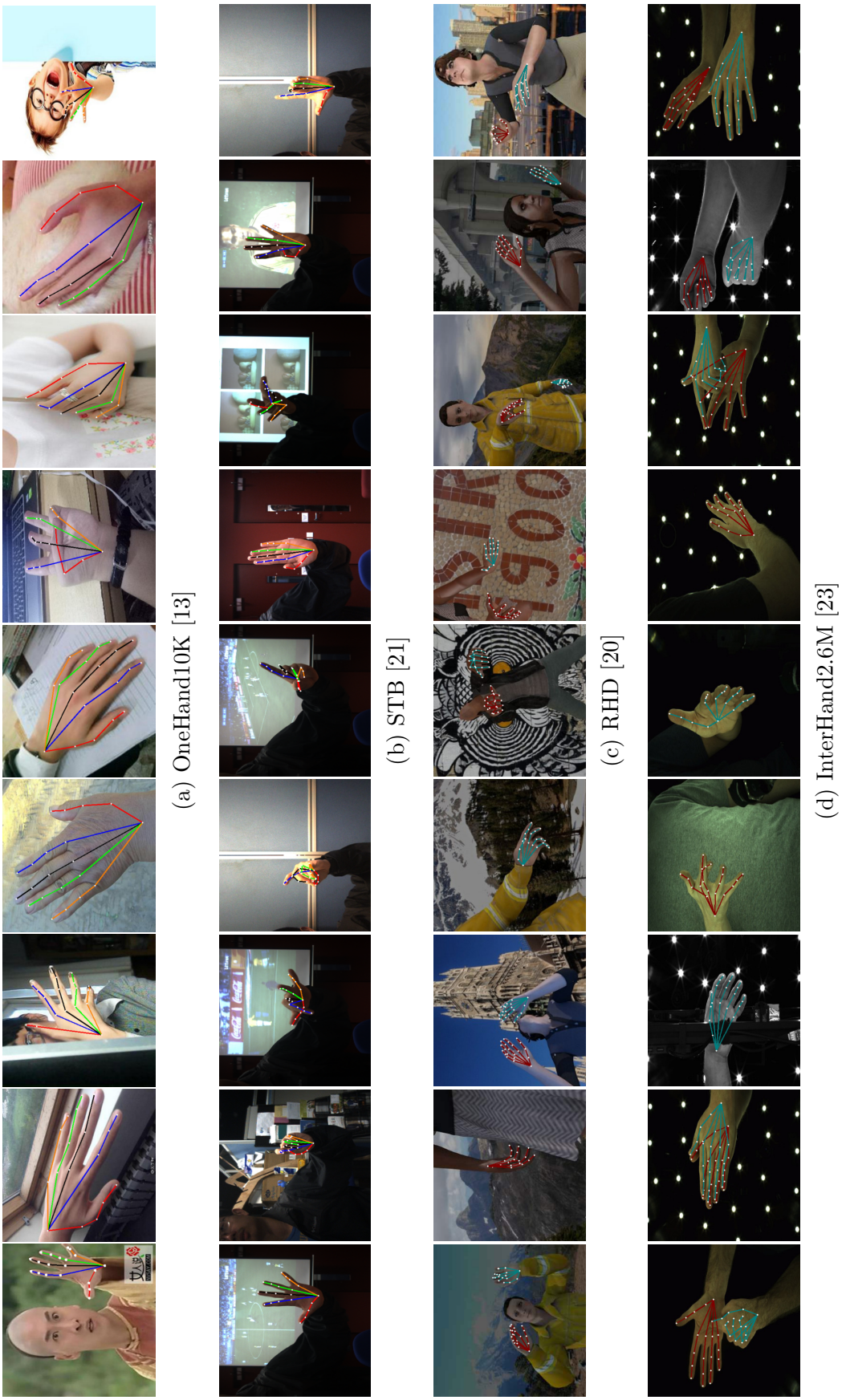


Figure 3.17: Quantitative analysis in terms of PCK values at different values of threshold distance. Separate analysis has been conducted for single and double-hand cases. (a) Single hand, and (b) double hands image samples from the RHD dataset. (c) Single hand, and (d) double hands samples for InterHand2.6M dataset.

Table 3.3: Hand Keypoints detection performance comparison on several existing datasets in terms of PCK

Models	Time (ms)	OneHand10K		STB		RHD				InterHand2.6M (H + M)			
						Single hand		Double hands		Single hand		Double hands	
		PCK ($\sigma = 0.2$)	mean	PCK ($\sigma = 0.2$)	mean	PCK ($\sigma = 0.2$)	mean	PCK ($\sigma = 0.2$)	mean	PCK ($\sigma = 0.2$)	mean	PCK ($\sigma = 0.2$)	mean
OpenPose [84]	73.44	0.5992	0.7446	-	-	0.8547	0.8701	0.7246	0.7693	0.8716	0.8754	0.6724	0.8003
Mask-Pose [13]	36.37	0.4473	0.6692	0.6738	0.8155	-	-	-	-	-	-	-	-
SRHandNet [14]	38.71	0.5415	0.7286	0.7721	0.8698	0.5438	0.8254	0.3432	0.6899	0.5191	0.8114	0.2596	0.6972
A2J [126]	32.75	0.1748	0.5408	0.2732	0.4858	0.4708	0.6108	0.1031	0.2669	0.8321	0.8635	0.2486	0.6569
SpatialNet [3]	60.74	0.5385	0.7046	0.4813	0.7685	0.6963	0.7272	0.4307	0.5281	0.4408	0.6698	0.4889	0.7322
Nearest-Neighbor (Proposed)	24.67	0.7151	0.7875	0.9483	0.9517	0.9578	0.9351	0.8236	0.8488	0.9498	0.9514	0.8453	0.8581



(a) OneHand10K [13] (b) STB [21] (c) RHD [20] (d) InterHand2.6M [23]

Figure 3.18: Sample results of hand keypoints detection from different datasets. For a single hand, the skeleton of each finger is shown with different colors. In double hand, the right-hand skeleton is shown in red color while the left-hand skeleton is shown in cyan color. The fingertips and finger joints are marked with white dots on the skeleton.

The quantitative comparison is conducted using (2.12). The performance analysis for single-hand hand keypoints detection is performed on OneHand10K [13] and STB [21]. We used full-size images for the proposed model's performance analysis. The results obtained are reported in Table 3.3. The proposed model is able to locate the keypoints in full-size images on test samples of these two datasets. For performance analysis on double hands, we used RHD [20], and InterHand2.6M [23] datasets. Since these two datasets have both single and double-hand image samples, we performed the analysis of those two separately. The PCK curves for single and double hands from samples of RHD [20], and InterHand2.6M [23] datasets are shown in Figure 3.17. The comparison in terms of PCK values at $\sigma = 0.2$ and averaged mean PCK values have been reported in Table 3.3. From the table, it can be inferred that the proposed method has better PCK values as compared to the aforesaid methods. The performance of the proposed method in hand keypoints detection is superior for both single and double hands. Some sample hand keypoints detection results are shown in Figure 3.18. As can be observed from the figure, the proposed model can estimate the keypoints' position in an RGB image with sufficient accuracy. The keypoints positions for single as well as double hands can be obtained with the proposed method even though the hand region is occluded.

3.4.7 Discussion

This section put forth a solution to the problem of 2D hand pose estimation from monocular RGB images. The results demonstrate that the hand keypoints can be directly located from a single full-size image. The method works effectively in real-world scenarios like illumination variations, complex backgrounds, different skin color types, etc. The performance of the proposed model is unaffected by different viewpoints, hand postures, and sizes. The network's speed is acceptable enough to be deployed in gesture-based applications such as air-writing, virtual typing, etc. The proposed model is designed to estimate the keypoints position from two hands simultaneously. However,

the current design is restricted only to two hands. The model might get confused if more than two hands are present in the image and may produce an incorrect hand keypoint location. It is also observed that the capability of the model to detect keypoints hidden from view is mainly dependent on the training data. The work proposed in this paper is limited to 2D hand keypoints position estimation in an RGB image. However, there are methods available that can use the 2D hand keypoints position to obtain 3D hand keypoints.

Most of the referred methods can estimate hand keypoints with insufficient accuracy on the full-size image. For example, the A2J [126] network was designed to work on cropped hand images; it does not perform well on a full-size image. Moreover, a keypoint location is predicted by aggregating the outputs of all anchor points. It causes the predicted location of a keypoint to deviate more from the ground-truth position. Its performance is also low, especially in the case of keypoints detection for double hands. Similarly, the OpenPose [84] was designed for an image wherein the hand has occupied a significant portion of the entire image. Its performance deteriorates when the hand's area is a small fraction of the entire image. It is so because it was designed with large output feature maps which cannot capture sufficient information from a small hand area in a full-size image. The Mask-Pose [13] model is dependent on hand mask generation, which is affected by conditions like illumination variation, complex background, etc.

Concluding Remarks

In this chapter of the dissertation, the process used for complete hand keypoints detection from a monocular RGB image is discussed. Most commonly, the detection and tracking of the positions of 21 hand joints are used for hand pose estimation. Similar to the process of partial keypoints detection, the two-step top-down approach was traditionally used for all 21 hand keypoints detection. Moreover, the heatmaps regression is used as either an intermediate or final step for hand keypoints detection. The deep learning methods are predominantly and more frequently used for hand keypoints

detection with the availability of a faster and more robust CNN-based algorithm for object detection from monocular RGB. This chapter covered two different approaches to hand keypoints detection. The first method was designed for single-hand keypoints detection and used a grid structure to locate the positions of keypoints in the image. Along, with that, a feedback inference mechanism was incorporated that helps improve the keypoints detection accuracy for small hands. The second approach for keypoints detection takes forward the grid-based approach for keypoints detection. The takes hints from the neatest-neighborhood pose particle-based process used for fingertips detection and a variation of it is used for complete hand keypoints detection. The second approach takes care of the simultaneous hand keypoints detection problem of both hands of a user. However, all these discussed approaches for the estimation of hand keypoints' position were limited to 2D hand pose estimation. Nevertheless, there are multiple approaches available that can take 2D keypoints positions and can estimate the 3D positions.