Chapter 4

Multiple Hands Keypoints Detection

4.1 Overview

The hand keypoints detection processes described in Chapter 2 and Chapter 3 focused on either single or double hands. Additionally, those algorithms were initially designed to work on a single hand and then extended to work on a double-hand configuration. The process of modification involves either replacing the architecture of the final layers with a new design or increasing the output neurons to accommodate the increased number of keypoints to be detected for both hands of a person. For example, the output layer of the CNN model used for hand keypoints detection with the process described in Section 3.4 is shown in Figure 4.1. Two groups (shown in red and green) of the same compositions are being used for hand localization and hand keypoints detection of both hands. This type of architecture is well suited only to those cases where using only a single hand or both hands is required. However, there are certain disadvantages associated with this type of approach. First and foremost, these types of architectures are rigid and cannot be easily modified to accommodate more additional hands. To perform keypoints detection of additional hands will require

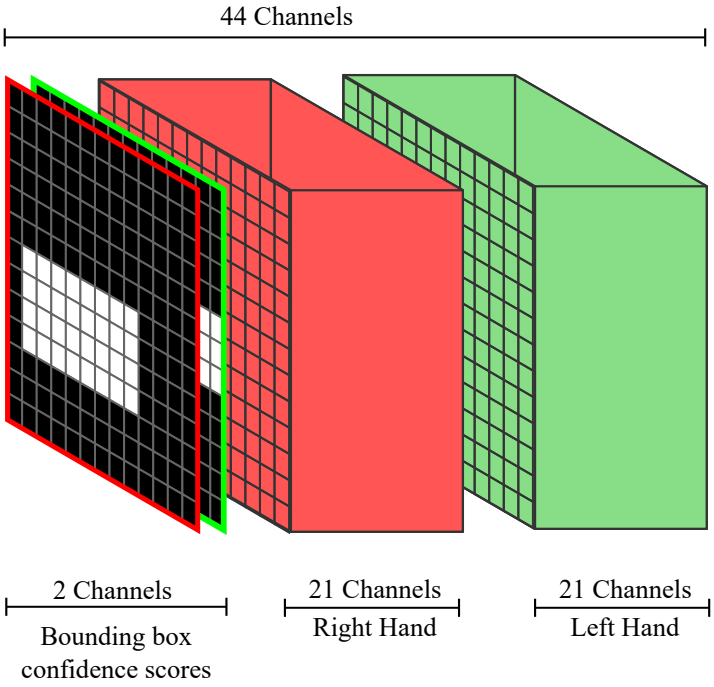


Figure 4.1: Increasing the number of output channels in the prediction layer to perform double hand keypoints detection.

changing the model architecture and retraining or finetuning which is a time-consuming process. Apart from this, the computational complexity of the model will increase, and thus the inference time may increase.

However, in the vision community, there has rapid development in object detection and semantic segmentation in a short period of time. This growth is visible after the availability of advanced architecture like Fast/Faster R-CNN [100, 142] and Fully Convolutional Network (FCN) [136] frameworks. These methods provide the capability of detecting each instance of an object being present in the input along with faster inference time. The problem of simultaneous multiple hands keypoints detection can be solved by considering each hand present in the instance as a separate instance. The process is referred to as instance segmentation. Once, all the instances of hand have been detected, a separate prediction layer processes each of these instances to estimate the positions of hand keypoints. This enables getting multiple hands keypoints positions simultaneously in one step.

4.2 Related Works

The multiple hand keypoints task can be considered analogous to multiple object detection. Here, the word object refers to the human hand. Deep convolutional neural networks (ConvNets) have significantly improved object detection accuracy. However, object detection is a challenging task as it first requires the identification of the object and locating the region in the image where it is present [143–145]. Also, there can be multiple instances of the same object in the image. The approach is to first generate proposals. Selective Search has been most popularly used for this purpose [146]. However, the Selective Search process is slower and it takes about 2 seconds per image in a CPU implementation. The solution to this proposed by Ren *et al.* [100] where the region proposal network (RPNs) which was initially developed on CPU is re-implemented for the GPU. The RPNS are designed to be efficient in predicting objects with wide ranges of scales and aspect ratios. To further take the object detection more scale invariant pyramids of filters are introduces [147, 148]. The Faster R-CNN [100] architecture advanced the attention mechanism with RPN. The architecture is flexible and robust and is the best-performing algorithm for object detection in several benchmarks.

The purpose of detecting each instance of an object called the instance segmentation process is based on the segment proposals. Methods like in [149, 150] resorted to bottom-up segments. DeepMask proposed segment candidates which were then classified using a separate CNN architecture. In this approach, segmentation precedes recognition, which is slow and less accurate. Another approach proposed by Dai *et al.* [151] used a multiple-stage cascade that predicts segment proposals from the bounding-box proposals. The process is then followed by a classification step. This multiple-stage cascading makes the overall process slower. Some other used semantic segmentation for instance segmentation. In the semantic segmentation task, the classification of objects takes place at pixel level [152–154]. In some of these approaches, the classified pixels are grouped to form different instances. However, a different approach is taken in the

Mask R-CNN [99] where instance-first strategy is employed.

In many of the processes used for hand keypoints detection, the hand detection is performed first [20, 50, 119]. For, the hand localization step in a color image several techniques have been proposed so far. Skin segmentation is one of the popular methods. Multiple methods have been proposed to create the hand mask [13, 20, 51, 155]. However, skin segmentation itself is a challenging task, and it is affected by factors like illumination, background, and skin-like objects in the image. An alternative to this is to perform bounding box detection by hand. The hand localization using bounding box regression is similar to an object detection task. But, it is a binary object detection (hand or background) task. Several researchers have retrained DNN-based object detection architectures for hand localization [48, 101, 119].

4.3 Methodology

The simultaneous detection of hand keypoints detection from multiple hands methodology takes inspiration from the two separate object detection algorithms namely the Mask R-CNN [99] and RetinaNet [4]. The CNN model used for this multiple hand keypoints detection can be broken down into two parts -

1. Hand ROI detection
2. Hand Keypoints detection

The hand ROI detection process is where from the proposal received from the region proposal network, the valid hand ROI will be selected. The hand ROI network can alsobe broken down into three components

1. Backbone model with feature pyramid pooling (FPN) [156]
2. Sub-network for object classification
3. Sub-network for object regression.

The architecture described above is shown in Figure 4.2. The backbone model architecture is composed of two paths - the bottom-up path and the top-down path. The bottom-up pathway is designed using the ResNet [40] architecture. It is used for the

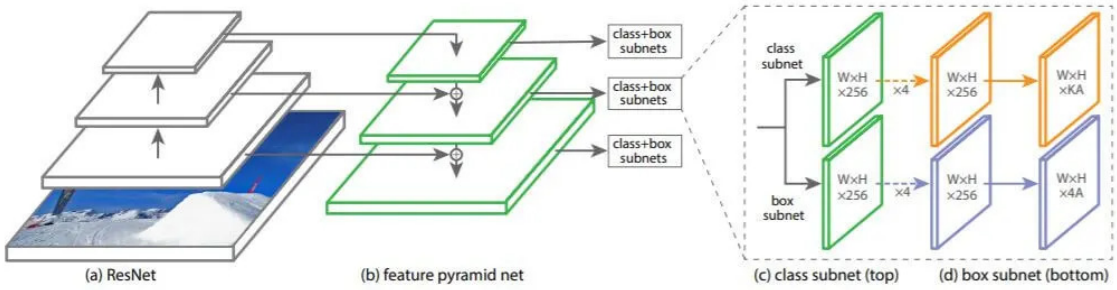


Figure 4.2: The architecture used for hand ROIs detection. Source [4]

extraction of features from the input image. In the design, the feature map at different scales is calculated. As we move from bottom to top in forward direction the spatial resolution of the features map is divided by a factor of two. The leftmost portion shown in Figure 4.2 represents the same. In the top-down pathway, the feature map received from the previous layers is sampled by a factor of two. In addition, there are lateral connections that merge the features of the top-down layer with that of bottom-up layers with the same spatial size.

At the high level, feature maps tend to have small resolutions though they are semantically stronger and thus more suitable to detect large objects. Just the opposite of this, the lower level feature maps are of high resolution and therefore are better suited to detect small objects.

The combination of the top-down paths and their lateral connections with the bottom-up pathways, which do not require extra computation, results in feature maps that can be both semantically and spatially strong. This allows the architecture to become scale-invariant and perform at a higher speed maintaining acceptable accuracy.

Next, there are two sub-networks in the architecture - sub-network for object classification and sub-network for object classification.

Sub-network for object classification: An FCN is attached to each feature map obtained from the top-down layer for object classification. The same is shown in Figure 4.2. The sub-network is composed of 3×3 convolutional layers with 256 filters.

Another convolutional layer with 3×3 kernel and $K \times A$ filters. Therefore, the output feature map has the size of $W \times H \times K.A$. Here, W and H are proportional to the width and height of the input feature map, respectively. The value of K and A signifies the number of objects and anchor boxes, respectively. If there are A number of anchor box proposals for each position in the feature map obtained from the last convolutional layer of the sub-network then the output feature map has the $K \times A$ channels.

Sub-network for object classification: The regression sub-network is also attached to each feature map obtained from the top-down layer. Design-wise both the sub-networks are identical except the last convolutional layer in the regression sub-network is of size $3\times$ with four filters. The resulting output feature thus has the size of $W \times H \times K.4$. The number 4 signifies the number of parameters produced for each anchor box. The regression sub-network predicts the relative offset in terms of the center points of the bounding box, its width, and height. Hence, the output feature map of this sub-net has $4 \times A$ channels.

With the aforementioned, the proposal for the hand ROI is obtained. The next step in the process is to filter bad proposals for which a non-maximum suppression algorithm is used [157]. The flow is shown in Figure 4.3 and it remains the same at the time inference as shown in Figure 4.4. Once the valid hand ROI proposal is obtained the feature making is performed. The process of feature masking is explained next.

Features masking. This process basically aimed at keeping the relevant features of the backbone layer and filtering out unnecessary information. In order to perform feature masking, a binary mask having the same spatial dimension is created using the ROI proposal. The ROI proposal layer provides the bounding box coordinate of the hand region in the input image. This bounding box is used to create the aforesaid binary mask. Next, logical AND operation of the feature map obtained from the backbone model with binary mask creation is conducted. The output of the AND operation is a feature map that has information in only those pixel locations where the binary mask

value was true. The rest of the pixel value in the feature map is made zero. The spatial dimension of the feature map remains the same as it was when taken from the backbone model. This flow is shown in Figure 4.3. Afterward, this resulting feature map from the feature masking process is used as the input source to the hand keypoints detection process.

The keypoints detection stage in this methodology can be either a heatmap regression process or a latent heatmap regression process. The keypoints detection methodology described in Chapter 3 can also be used in the keypoints detection stage.

4.3.1 Training Pipeline

The flow diagram shown in Figure 4.3 highlights the processes involved in the training of this proposed deep neural network architecture. The CNN model used in this design can be trained end-to-end. The complete flow has two parts each with its own objects. The first part is performing the hand ROI detection and the next part is performing the keypoints. Hence, to train the model two separate loss function is used. At training time the model has three loss components in total components.

$$loss_{total} = loss_A + loss_B \tag{4.1}$$

where,

$$loss_A = L_{smooth} + L_{focal}, \tag{4.2}$$

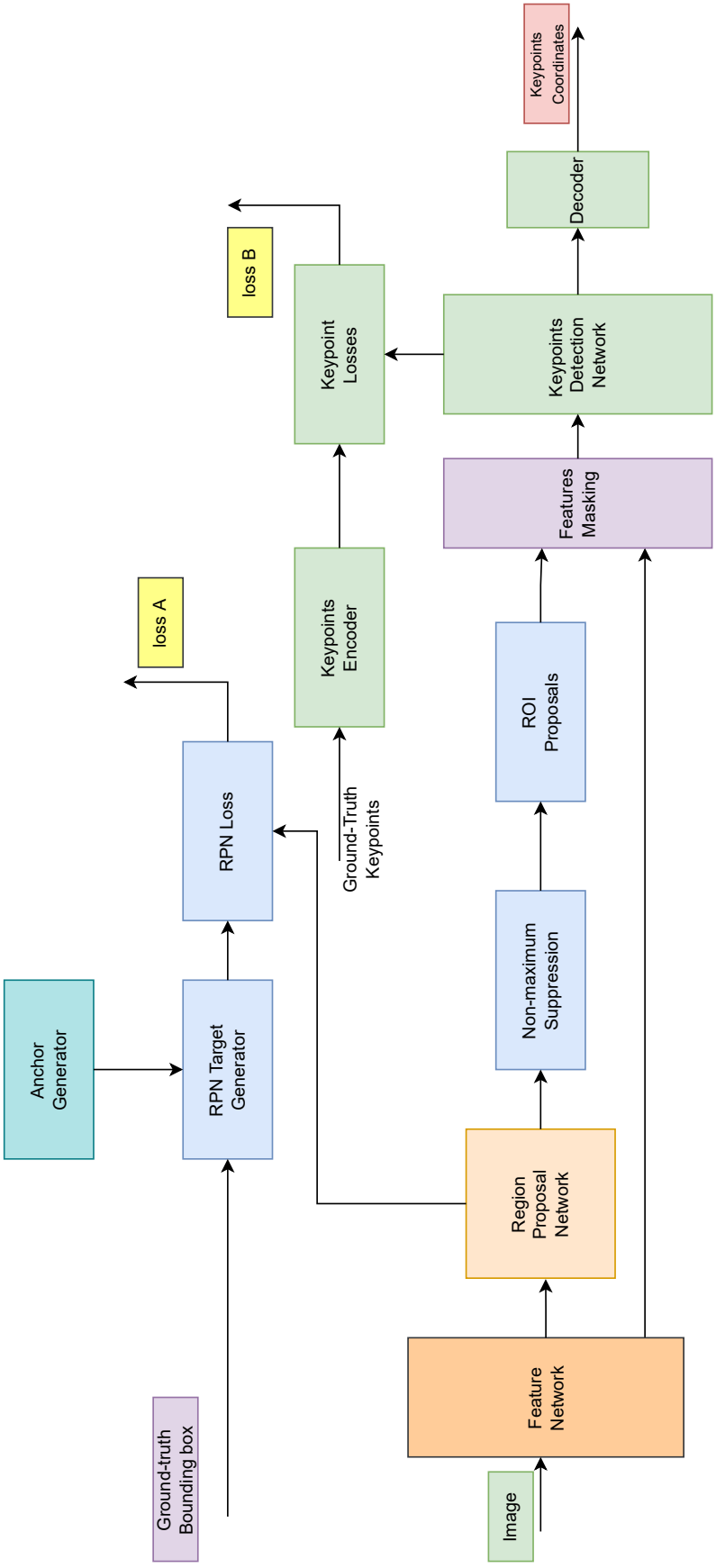


Figure 4.3: The flow diagram for training pipeline.

and

$$loss_B = \sum \hat{M}.(\hat{y} - y)^2 + \sum (1 - \hat{M}).(\hat{y} - y)^2, \tag{4.3}$$

.

In (4.2), L_{smooth} is the smooth loss given by

$$L_{smooth} = \begin{cases} |x|, & \text{if } |x| > \alpha \\ \frac{1}{|\alpha|}.x^2, & \text{if } |x| \leq \alpha \end{cases} \tag{4.4}$$

and

$$L_{focal} = \begin{cases} -\alpha(1-p)^\gamma \log(p), & \text{if } y = 1 \\ -(1-\alpha)p^\gamma \log(1-p), & \text{otherwise} \end{cases} \tag{4.5}$$

The value of $\hat{M}$ in (4.3) is calculated as

$$\hat{M} = \frac{\hat{y}}{\hat{y} + \epsilon}, \tag{4.6}$$

The SGD optimizer is used to optimize the parameters of the model. The model was trained by combining the samples from OneHand10K [13], RHD [20], and InterHand2.6M [23]. The OneHand10K is composed of samples taken in the natural environment of a single hand. The RHD and InterHand2.6M datasets are composed of samples from multiple hands.

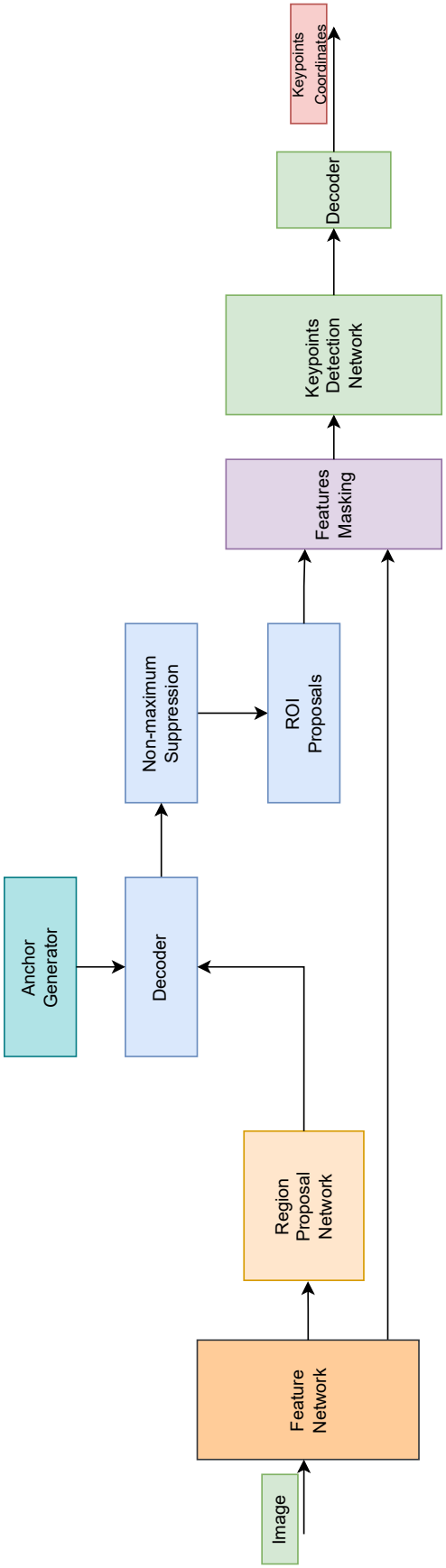


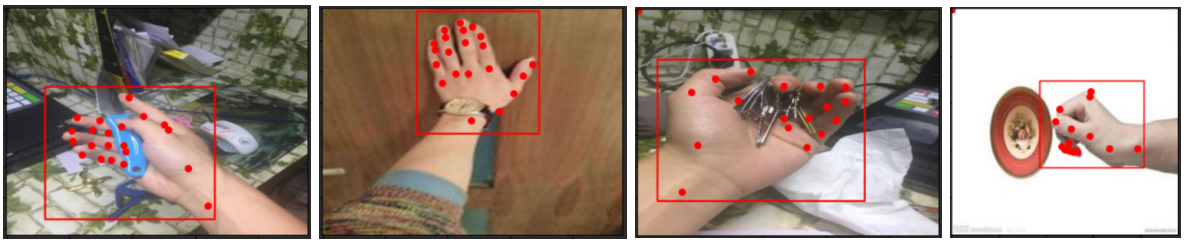
Figure 4.4: The flow diagram for inference pipeline.

4.3.2 Inference Pipeline

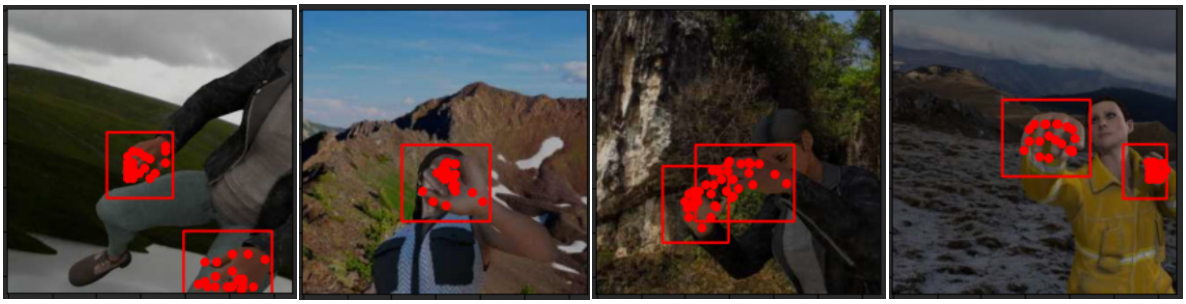
The flow diagram present in Figure 4.4 shows the process followed at the time of inference. The input to the mode is a full-size monocular RGB image. The image is first passed to a feature extractor layer. The feature map **F**, from the feature extractor, is then used by the RPN layer to localize all the hands present in the image. This detection takes the help of anchor boxes. The anchors' boxes are predefined bounding boxes with fixed dimensions and aspect ratios. A hand-bounding box is generated using one such anchor box and adding an offset to it which is calculated by the RPN. Once, the hand bounding box is available, the process of feature masking takes place. The resulting features are then used by a keypoints detection algorithm to give the position of keypoints for all the hands present in the input image.

4.4 Results

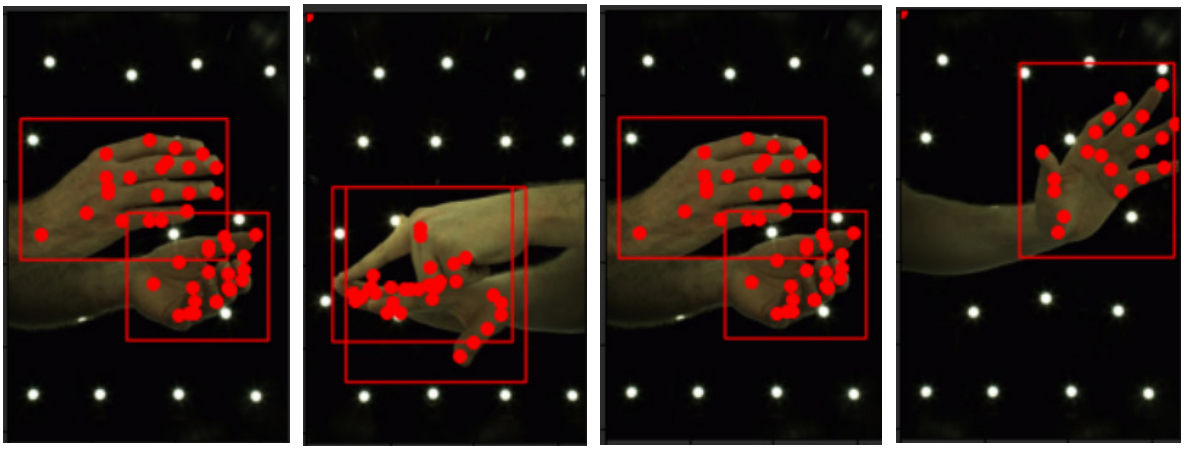
The inference results obtained on the samples from the validation set of three aforementioned datasets are shown in Fig 4.5. The red points denote the hand joints' position in the image. The bounding boxes are shown in red. It can be observed from the figure that the proposed model is available to locate both the hands of a user even if it is partially occluded. These results show that the model is able to understand the contextual information. The comparative analysis is conducted using PCK which is defined in (2.12). The results are furnished in Table 4.1. The SRHandNet, A2J, and Iqbal *et al.*algorithms were used for comparison. These architectures are trained on the aforementioned datasets for a fair comparison. Along with that, for the keypoints detection process, three different processes namely heatmaps regression, latent heatmap regression, and grid-based process (described in Section 3.4) are used. The latter model is the best performing among the methods used in comparison. While rest of the methods or algorithms were not designed for keypoints detection and their structures are rigid. This made it difficult to adopt them for multiple hand keypoints detection. However, the proposed architecture is able to detect keypoints for any number of hands given



(a) OneHand10k



(b) RHD



(c) InterHand2.6M

Figure 4.5: The sample results of hand ROI and keypoints detection

their scale is appropriate and the hands are not occluded.

Concluding Remarks

The most common approach to performing an estimation of the keypoints' position is to use the top-down approach. First, the subject of interest, for example, in the task of hand keypoints detection, the localization of the hand takes place at first. Then the keypoints are detected using a separate and independent methodology. The main

Table 4.1: **Performance comparison of the proposed multi-hand keypoints detection with three different keypoints detection algorithms and other deep learning-based methods**

Algorithms		OneHand10K		RHD		InterHand2.6M	
		PCK@0.2	mean	PCK@0.2	mean	PCK@0.2	mean
SRHandNet [14]		0.7221	0.74	0.876	0.8547	0.9267	0.9101
A2J [126]		0.7539	0.7512	0.8752	0.8636	0.9319	0.9261
Iqbal *et al.* [132]		0.7714	0.7633	0.8914	0.8839	0.9458	0.9398
Heatmaps regression	Proposed	0.7101	0.7333	0.8835	0.8726	0.9312	0.9483
Latent heatmap		0.7347	0.7621	0.9010	0.8823	0.9471	0.9514
Grid-based		**0.7956**	**0.7711**	**0.9165**	**0.8941**	**0.9561**	**0.9549**

drawback of this approach is the dependency of the keypoints detection process on the hand localization step. Additionally, the additional computational resources are confused, and the overall inference time of the model increases. The two-step approach is reduced to a single-stage process by refusing the intermediate features of a CNN model. As the hand ROI information is already extracted at the time of keypoints detection, the same has been used and all the possible hand regions are extracted. Then using parallel processing the keypoints detection is performed on a single hand. The advantage of this approach is that it makes the model flexible enough to work on a variable number of hands. Moreover, the computational resources and inference time are saved. Since the model is already learning the contextual information, the model prediction accuracy also increased.