IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 13 February 2024, accepted 8 March 2024, date of publication 11 March 2024, date of current version 22 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3376426

RESEARCH ARTICLE

Multiple-Hand 2D Pose Estimation From a Monocular RGB Image

PURNENDU MISHRA AND KISHOR SARAWADEKAR, (Senior Member, IEEE)
Department of Electronics Engineering, Indian Institute of Technology (BHU) Varanasi, Varanasi 221005, India
Corresponding author: Kishor Sarawadekar (skishor.ece@iitbhu.ac.in)

ABSTRACT Deep learning models and algorithms facilitate relatively easier ways of hand pose estimation from monocular RGB images compared to traditional approaches. Despite this, a majority of available algorithms use multiple-stage models to perform hand pose estimation. Moreover, the single-stage methods are mainly limited to a single hand and it is difficult for them to scale to multiple hands. To this end, we propose an approach that takes the features of the saliency map extracted for hand region of interest (ROI) localization. An integrated network uses these features for pose estimation. This arrangement of layers forms an end-to-end pipeline that allows simultaneous pose estimation for multiple hands. The model is designed to run on multiple cores of CPU/GPU to independently perform inference for each detected hands' pose making possible faster inference and hence suitable for real-time applications. In addition, a new approach using grid-based design to estimate hand-keypoints position with high precision is also proposed. Both the proposed designs are validated on multiple datasets to prove their feasibility and effectiveness. The probability of the correct keypoint (PCK) value at threshold value of 0.2 is above 95% on the test sets from Interhand dataset and Rendered HandPose Dataset (RHD).

INDEX TERMS Computer vision, convolutional neural networks, deep learning, hand pose, monocular, pose estimation.

I. INTRODUCTION

Hand pose estimation (HPE) is the process of modeling the human hand as a set of some parts (e.g. wrist point, fingertips, fingers' joints) and finding their positions in a hand image (2D estimation) or the simulation of hand parts positions in a 3D space. It is used to estimate hands with the phalanges and is the basis of dynamic gesture recognition. With the availability of low-cost and portable image acquisition devices, HPE offers a variety of uses. It can be used as a contact-less controller for human-computer interaction (HCI) and can have applications in the field of Sign Language Recognition (SLR) [1], [2], [3], activity recognition [4], Augmented Reality (AR), Mixed Reality (MR), Virtual Reality (VR) [5], [6], hand gestures recognition [7] (HGR) and others.

The associate editor coordinating the review of this manuscript and approving it for publication was Gianluigi Ciocca.

Usually, the hand pose is defined using 21 hand key-points [8], [9]; and those are 5 fingertips, the centers of 15 finger joints, and the base of the palm (wrist point). However, attaining high performance in a vision-based markerless hand pose estimation is an extremely challenging problem, especially from the perspective of color images. A human hand has a high Degree of Freedom (DOF) [10], which facilitates flexible hand and finger movements creating a challenge to estimate hand pose. Appearance ambiguities, self-occlusions, and occlusion due to objects in the image are additional factors that must be dealt with appropriately for an accurate hand pose estimation.

With the introduction and continuous improvements in deep learning (DL) algorithms, many computer-vision (CV) problems which were very challenging to solve with traditional features engineering techniques, became solvable. Different DL-based algorithms have been proposed for HPE. Many of them perform very reliably. In terms of the approach

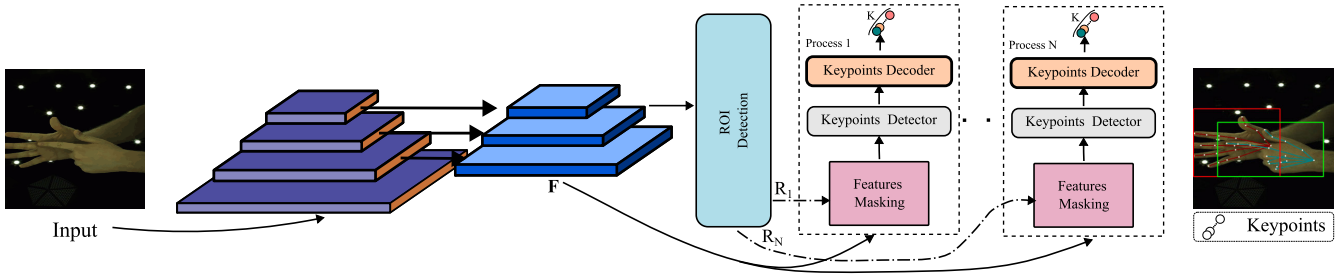


FIGURE 1. The complete flow diagram for simultaneous multiple hand keypoints detection process. The features *F* are used for ROI detections and keypoints detection. Parallel processes (referred to as Process N) perform keypoints detection for each detected hand region in the given monocular RGB image. The red and green bounding boxes represent the ROIs of each detected hand.

used to solve the HPE problem, the pose estimation methods can be broadly categorized into either top-down or bottom-up approaches. While the former approach depends on a separate process to detect and segment the hands before estimating the position of the hands keypoint, in the latter the joints are detected first before grouping them to form a unique hand pose [11], [12]. The top-down approach is the predominately used approach for hand pose estimations [1], [8], [9], [13], [14]. One major advantage of the top-down approach is its scale invariance pose estimation. However, this approach has a few downsides like dependence on a hand localization process and relatively slow estimation time due to the involvement of multiple independent stages in the inference pipeline.

The work presented in this paper is focused on simultaneous pose estimation for both hands of a single user. The methodology proposed in this paper is developed for 2D hand pose estimation from monocular RGB images. The goal is to perform hand pose estimation directly on the image captured from a conventional RGB camera and does not involve any independent hand localization step. The idea presented in this paper is inspired by the work of Wang et al. [8] in which they used a cascaded network for hand mask segmentation and estimation of hands' keypoint position. On analysis of the saliency map of the first stage network proposed by Wang et al. [8], it was observed the network first localizes the hand region and extracts relevant features which were then used by the subsequent network for pose estimation. Taking a cue from the aforementioned approach a network is designed that will localize hand region of interest (ROI), which will be then utilized to extract features from the saliency map. Using these extracted features, the hand pose will be estimated. The complete proposed process is illustrated in Fig. 1. Using the features extracted from a deep learning architecture, ROIs are detected and the same features are reused to perform hand-keypoints position estimation. The details of Fig. 1 and the proposed methodology will follow in Section III. The advantage of the proposed approach is the whole process is end-to-end and does not need any pre-processing step before HPE. Additionally, the keypoints position of multiple hands can be determined simultaneously.

The main contributions of this paper are summarized below.

1) An end-to-end pipeline is proposed for simultaneous hand pose estimation from multiple hands directly from a monocular RGB image.
2) The proposed design is modular enabling different pose estimation algorithms like heatmap regression, latent heatmap regression, or any other to be easily integrated in the pipeline.
3) A novel technique for hand pose estimation is also proposed that estimates keypoints'position using a nearest-neighbor estimation approach. The proposed design uses a generic template and is useful for hands of varying scales.
4) The proposed algorithm is validated on multiple datasets that provide annotated hand keypoints for both of the hands of a user.
5) The algorithm can run in parallel using multiple CPU/GPU cores of a system enabling it to have a faster inference time.

The organization of the rest of the paper is as follows. A brief study of related work is furnished in Section II. Section III details the proposed method. Experimental details are given in Section IV. The results analysis and comparison are presented in Section V. Section VI concludes the paper.

II. RELATED WORK

Due to its dexterous nature, the human hand has been used as an interactive tool in various HCI applications. Many efforts have been put in to capture the hand motion and interpret its meaning for interactive purposes. A survey of various methods available for estimation of hand pose had been conducted by Erol et al. [5], and Doosti [6].

The data gloves [15], [16], [17] are effective in capturing the hand motion. The hand and position of its keypoints can be monitored in real-time with high accuracy. However, these devices are expensive, restrict hand motion, and require complex calibration before pose estimation. Computer-Vision (CV) provides alternative solutions to the data gloves method for contactless pose estimation. 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 [18], [19], [20], [21]. However, a depth sensor is effective only in an indoor environment, and the depth-maps are noisy [1], [9], [22], [23]. Thus, the scope of a depth sensor is limited. RGB images present an alternative to depth-map for hand pose estimation. The proposed work is focused on 2D hand pose estimation from an RGB image. The existing work on hand pose estimation from RGB images is briefed in the next sub-section.

A. 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 [24] 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. [25] and Baydoun et al. [26]. The method in [25] used a hierarchical regression technique whereas, a graphical approach is followed in [26]. Grzejszczak et al. [27] has used distance transform and templates for hand keypoints detection from an RGB image. With the advent of convolutional neural network (CNN), considerable progress has been made in the field of hand pose estimation. A method using CNN for 3D hand pose estimation from an RGB image has been proposed by Zimmermann and Brox [1]. They used three different CNN networks to estimate the 3D position of hand joints. They used *PoseNet* network for 2D pose estimation from the cropped hand image. The PoseNet architecture produces a heatmap for each hand joint. The 2D coordinates of hand joints are then lifted to 3D coordinates using a *PosePrior* network. However, the PoseNet was trained with synthetic hand images and performs well on a synthetic dataset. However, similar performance is not observed in real-world images. Simon et al. [13] proposed a multiview bootstrapping method to generate a large annotated dataset. They trained a weak detector to generate hand annotation in a multiview situation. The detector is not robust, and the dataset is limited to a controlled environment. Recently, Wang et al. [8] proposed a two-stage cascaded CNN architecture for hand pose estimation from a single RGB image. They performed hand mask prediction in the first stage. The second stage uses the output feature maps as well as the hand mask generated in the first stage to estimate the hand joints'position. The second stage performs heatmap regression and it is based on *Convolutional Pose Machine* (CPM) architecture [28]. The authors also published a dataset called OneHand10K for 2D hand pose estimation from color images. This dataset comprises images from real-world scenarios. Another method called *SRHandNet*, was proposed by Wang et al. [9] for hand pose estimation from color images. The authors use an encoder-decoder architecture to perform heatmap regression to obtain hand keypoints position and corresponding bounding box coordinates. The bounding box is used to crop the hand from the color image and perform cycle inference through the same network to improve the hand pose estimation. Most of the aforesaid methods are dependent on hand localization. To eliminate the dependency on hand localization, Li et al. [23] proposed a hand keypoints detection method using pose anchors (a form of hand template) from a full-size image. This limits the hand keypoints detection to a few hand poses that match the template. Spatial Information Aware Graph Neural Network [29] is also used to extract the relation between joints and their neighbors but multi-scale information is not captured with this approach. HandyPose [30] architecture used multi-level features with waterfall atrous spatial pooling for 2D hand pose estimation.

B. MULTIPLE-HAND POSE ESTIMATION

It is defined as the estimation of the pose for all of the hands present in the input image in one go. The estimation of hand pose for multiple hands has received lesser attention in comparison to single-hand pose estimation. Wang et al. [9] used a hand ROI detector and performed pose estimation on all detected hands. Li et al. [23] used pose-anchors obtained after Object Keypoints Similarity (OKS) based clustering to determine the keypoints. These anchors can be used to detect keypoints for multiple hands. While the methods proposed by Wang et al. [9] can be categorized as a top-down approach, the method proposed by Li et al. [23] is a single-step approach.

The HPE process can be considered analogous to human or animal pose estimation. While in hand pose estimation, the pose is defined by 21 keypoints on a hand, the number of keypoints in other pose estimation processes varies. Here, we describe briefly a few human-pose estimation processes that can be adopted for hand pose estimation after making a few modifications. In general, the pose estimation approaches can be summarized either as top-down methods or bottom-up methods. Recently, single-stage methods were also proposed, a brief of related works using this approach is provided later in the Section.

1) TOP-DOWN METHODS

In this method, a detector is used to obtain bounding boxes of all the object instances present in the input image. Next, the pose estimation is performed for each of the detected instances of the object. Some work following this approach includes Hourglass [31], CPM [28], SimpleBaseline [32], HRNet [33],Regional Multi-Person Pose Estimation (RMPE) [34] and others. The top-down approach is dependent on the performance of the detector.

2) BOTTOM-UP METHODS

As defined in Section I, bottom-up methods first determine the position of keypoints and then they are grouped into individual instances in the input. The major focus of most of the bottom-up methods is on how to associate a keypoint to a group or instance. OpenPose [11] used affinity field while Part Association Field (PAF) is used in PifPaf [35]. Shi et al. [36] uses transformers for pose estimation. The

grouping process makes the bottom-up methods complex and their performance is often inferior to the top-down methods.

3) SINGLE-STAGE METHODS

As the name implies, the methods in this category try to map keypoints position to that of the object instances. The keypoint positions are obtained with regression. Some of the methods in this category includes, CenterNet [37], FCPose [38], InsPose [39], SPM [40] and so on. There is a trade-off between accuracy and efficiency in the single-stage methods. Even though some of these methods have performance equivalent to methods using the top-down approach, not all of them are end-to-end optimized and require some form of pre-processing.

In this work, we present an end-to-end optimized single-stage network that shares the saliency map across sub-networks for HPE. The saliency map sharing helps in reducing the number of parameters in the model. Apart from this the contextual information captured from a full-size image as input helps apprehend the spatial and semantic relationship between multiple hands. Overall, this feature sharing helps to differentiate multiple hands along with capturing the affinity field of the different joints of the hand making the design more consistent and precise.

III. METHODOLOGY

A. OVERVIEW

Our goal is to find the 2D position of the hand joints from a single RGB image that may contain multiple hands. In this work, our focus is especially on the two hands of a single user who is interacting with a computer. Our method consists of two main steps: detecting the regions of interest (ROIs) where the hands are located and estimating the keypoints of each hand within the ROIs (similar to the top-down approach). The ROI detection is an intermediate and a hidden stage. This means that the hand detection results are only utilized to perform subsequent keypoints detection processes and these results are not available at output. Additionally, the features used for ROI detection are re-used for keypoints detection. The input to the algorithm is a monocular RGB image $I \in \mathbb{R}^{w \times h \times 3}$. The algorithm attempts to estimate the position of a set of K hand joints (of a single hand) in 2D space. The location of a keypoint is represented by point $p_n = (x_n, y_n) \in \mathbb{R}^2$ where $n \in [1, K]$. Typically, for a single hand, the value of K is 21. The following sub-sections provide the details of the processes used to estimate the positions of hand keypoints starting with ROI(s) detection.

B. MULTIPLE HAND POSE ESTIMATION

The simultaneous estimation of hand keypoints' position from multiple hands methodology takes inspiration from state-of-the-art object detection algorithms. The CNN model used for this multiple hand keypoints detection can be broken down into two steps - 1) Hand ROI detection 2) Hand Keypoints detection.

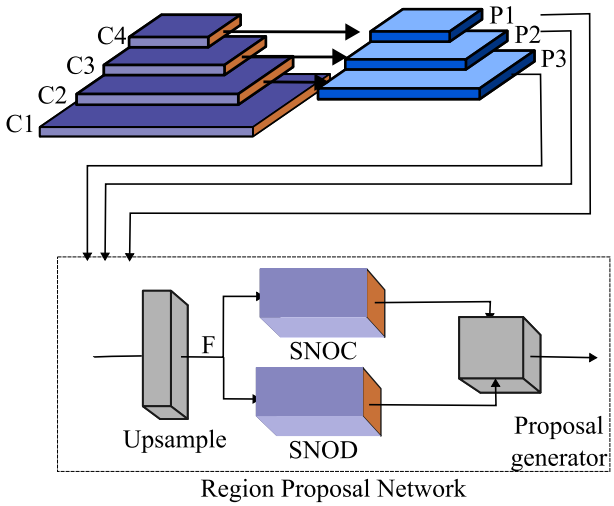


FIGURE 2. Flow diagram representing the hand ROIs proposal network. The classification sub-net distinguishes between foreground and background. The bounding box (bbox) regression sub-net provides coordinates of the foreground.

A single end-to-end pipeline, shown in Fig. 1, is designed to perform both of these two steps. Firstly, from the input image, ROIs are selected, and using these ROIs keypoints for each individual detected hand are localized. Roughly, the CNN architecture, shown in Fig. 1, can be divided into three parts:

1) Backbone model with feature pyramid pooling (FPN) [41], 2) ROI detection network, and 3) keypoints detection network.

The feature maps F from the backbone model are shared between ROI detection and the keypoints detection network. The architecture of ROI detection along with the backbone model is shown in Fig. 2. The backbone model architecture is composed of two paths - the bottom-up path and the top-down path. Layers $C1 - C4$ constitute the bottom-up path. As we move from bottom to top in the forward direction of the bottom-up pathway, the spatial resolution of the features map is divided by a factor of two. The leftmost portion shown in Fig. 2 represents the same. Layers $P1 - P3$ form the top-down pathway. These layers are a combination of upsampling (bilinear interpolation) and convolutional (kernel of size 3×3) layers. The layers in top-down pathways are connected to layers from bottom-up pathways using skip connections. For the bottom-up pathway multiple state-of-the-arts deep learning architectures like ResNet [42], DenseNet [43], and EfficientNet [44] are tried. The backbone model is used as a feature extraction workhorse of the model. These pathways in the backbone model are designed considering the following. 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.

After the backbone model, the next part of the architecture is ROI detection. The input to the ROI detection portion of the network are features obtained after adding features $P1$, $P2$, and $P3$ as shown in Fig. 3. Before adding these features, they are upsampled to the same spatial resolution. Features P1 and P2 are upsampled 4 and 2 times using bilinear interpolation, respectively. This upsample feature is referred to as F and has been used for ROI detection as well as keypoints detection as shown in Fig. 1.

For the ROI detection process, two small sub-networks are used. One of the sub-networks is used for object classification called SNOC (sub-network for objection classification) and the other sub-network is used for hand bounding box detection. It is called a sub-network for objection detection (SNOD). The details of the two sub-networks are as follows.

1) SUB-NETWORK FOR OBJECT CLASSIFICATION (SNOC)

A fully convolutional network (FCN) is attached to feature map F obtained from the upsampling layer for object classification. The same is shown in Fig. 2 and Fig. 3. The sub-network is composed of one 3×3 convolutional layers with 256 filters. Another convolutional layer attached sequentially to it has 3×3 kernel and $O \times A$ filters. Therefore, the output feature map has the size of $W \times H \times (A \cdot O)$. Here, W and H are proportional to the width and height of the input feature map, respectively. The value of O and A signifies the number of objects (hands) 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 $A \cdot O$ channels. These feature maps are then reshaped to form a 2-D array of shape $(W \cdot H \cdot A) \times O$. This array is then passed through a sigmoid activation (applied to the last axis). The sigmoid layer ensures that each object classification is mutually exclusive to the other, for all of the anchors. The sigmoid-activated array is used with the Non-Maximum Suppression (NMS) [45] algorithm to generate ROI proposals.

2) SUB-NETWORK FOR OBJECT DETECTION (SNOD)

Similar to SNOC, the feature map is used for ROI bounding box coordinates regression. 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 $W \times H \times (4 \cdot O)$. 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 $4 \cdot A$ channels. These feature maps are then

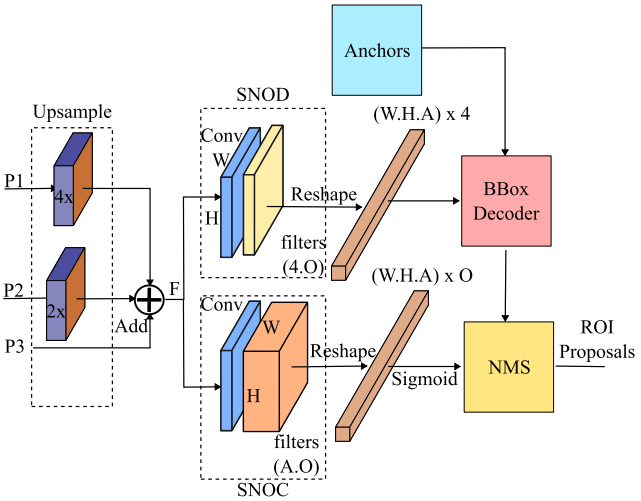


FIGURE 3. The region proposal network of ROI proposal generation.

reshaped to a 2-D array of shape $(W \cdot H \cdot A) \times 4$. This array is then combined with the anchors to get actual ROI coordinates as explained below.

3) ROI BOUNDING BOX DECODING

The prediction of sub-network for object detection are offset values with respect to the anchor boxes are presented as $[\Delta x, \Delta y, w', h']$. Here, $(\Delta x, \Delta y)$ are the offsets value w.r.t. centroids of the anchors (x_{anchor}, y_{anchor}). And, (w', h') are the logarithmic value of the ratio of the objects'bounding box width and height to the anchors'width and height, respectively. The predicted bounding box $[x_{roi}, y_{roi}, w_{roi}, h_{roi}]$ with the corresponding anchor box $[x_{anchor}, y_{anchor}, w_{anchor}, h_{anchor}]$ can obtained as

$$x_{roi} = \Delta x \sigma_x w_{anchor} + x_{anchor} \quad (1)$$

$$y_{roi} = \Delta y \sigma_y h_{anchor} + y_{anchor} \quad (2)$$

$$w_{roi} = \exp(w' \sigma_w) w_{anchor} \quad (3)$$

$$h_{roi} = \exp(h' \sigma_h) h_{anchor} \quad (4)$$

Above $\sigma_x = 0.1$, $\sigma_y = 0.1$, $\sigma_w = 0.2$ and $\sigma_h = 0.2$ are the standard deviation values used for normalization. The normalization with standard deviation is used to achieve better DL accuracy.

4) ROI PROPOSALS

The predicted ROI confidence scores and its bounding box coordinates $[x_{roi}, y_{roi}, w_{roi}, h_{roi}]$ are passed as input to the NMS algorithm to get the probable hand (ROI) proposals. The flow is shown in Fig. 5 and it remains the same at the time of inference as shown in Fig. 6. Once the valid hand ROI proposal is obtained the feature masking is performed. The process of feature masking is explained next.

5) FEATURES MASKING

This process keeps the backbone layer's important features and discards the unnecessary ones. The aim is to keep

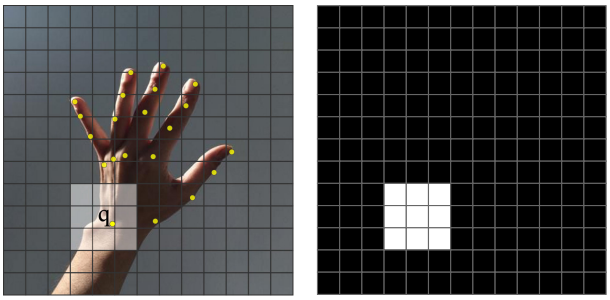


FIGURE 4. An illustration of the method used for the estimation of 2D hand keypoints position from an RGB image.

the features that are useful for keypoints detection without altering their spatial dimension. For this, a binary mask with the same spatial size as the ROI proposal is made. The ROI proposal layer gives the hand region's bounding box in the input image. This bounding box is used to make the binary mask. Next, the feature map from the backbone model and the binary mask are merged with a logical AND operation. The outcome is a feature map F' that only has information where the binary mask is true. The other pixels in the feature map are set to zero. The feature map's spatial size does not change from the backbone model. This flow is shown in Fig. 5. Afterward, this resulting feature map from the feature masking process is used as the input source to the hand keypoints detection process. The architecture, shown in Fig. 1, is designed to be modular so that different keypoints models can be easily plugged into the proposed architecture. So, in the proposed architecture, methodologies based on either a heatmap regression process [8], [9] or a latent heatmap regression process [46] can be used. In addition to these most commonly used methods for keypoints position estimation, a new method using a grid-based structure using a nearest-neighbor probability-based estimation process is proposed. The details of which are provided in Section III-C.

C. HAND KEYPOINTS DETECTION

A new method for estimation of hand keypoints proposed in this work is based on nearest-neighbor pixels around a keypoint location. The details of the methods are as follows. The nearest-neighborhood-based method is inspired by how human experts annotate keypoints. If different experts mark keypoints on their own, they will not agree exactly. But they will be close to each other (these are the neighborhood points). Also, the true keypoint will be near the center of these points. Based on this hypothesis, we formulate the process of keypoints position estimation as the process of detection of N neighborhood points. We calculate the probable position of the hand keypoint using these neighborhood points.

We divide the input features into $M \times M$ square grids where the value of M is 32. This grid structure helps in locating hand keypoints in it. Predictions are made using a CNN-based architecture. The proposed CNN architecture takes these features as input and produces a three-dimensional saliency 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. As an example, the wrist point marked as 'q' as shown in Fig. 4a is encoded to the pattern formed by white grid cells shown in Fig. 4b. 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 $q = \{q_m\}_{m \in M}$, 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 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 cell) are assigned the value of one, and the remaining cells'values are made zero. The white cells in Fig. 4b represent the center and eight neighboring cells for the wrist point of the hand shown in Fig. 4a. This pattern is formed for all K keypoints. We create an array of shapes $M \times M \times K$ during the encoding process.

1) KEYPOINTS DETECTION NETWORK

A small network consisting of three CNN layers and rectified linear unit (ReLU) is used for keypoints detection. All of these layers have a kernel of size 3×3. The first layer has 256 filters whereas, the second layer connected sequentially to it has 128 filters. The last layer is the output layer. The number of filters in this layer is equal to K. All of these layers are wrapped around the *"Time-Distributed"* layer. This allows the application of the same convolution operation to each instance of the input.

2) KEYPOINTS POSITION DECODING

At the time of making the inference, we get a feature map of shape $M \times M \times K$ (same as the encoded array defined in the above paragraph). Each cell in the output array represents a probabilistic value in the range [0, 1]. We use a threshold value of $\delta_q \in [0, 1]$ to find all such grid cells that possibly represent a group of cells where a keypoints is possibly present. In addition, we add one extra constraint of a minimum of two cells that must have a value above δ_q and these cells should be adjacent. For all such neighbor grid cells having a value above the threshold, $\delta_q = 0.5$,

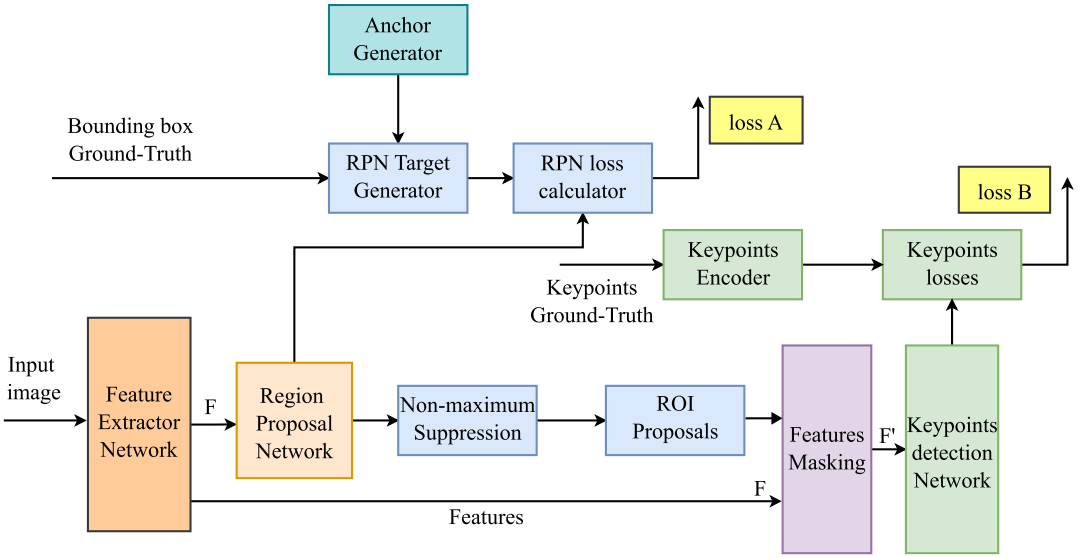


FIGURE 5. The flow diagram for training pipeline. The hand ROI detection and pose estimation networks are integrated with an end-to-end pipeline.

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_u, y_v) can be calculated as

$$x_u = (j + 0.5)\frac{s}{M}, \tag{5}$$

$$y_v = (i + 0.5)\frac{s}{M}. \tag{6}$$

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 input image is given by s and in our experiments the value of $s = 512$ pixels. The final estimated keypoint coordinate is the centroid of all points obtained during the decoding process. And, this final keypoint coordinate $\hat{p}_n = (\hat{x}_n, \hat{y}_n)$ is calculated as

$$\hat{x}_n = \frac{1}{N}\sum_{u}^{N} x_u, \tag{7}$$

$$\hat{y}_n = \frac{1}{N}\sum_{v}^{N} y_v. \tag{8}$$

Here, x_u and y_v are coordinates estimated in (5) and (6), respectively.

D. TRAINING PIPELINE

The flow diagram shown in Fig. 5 highlights the processes involved in the training of this proposed deep neural network architecture. The process of getting the ROI proposal and estimating the keypoints'position is an end-to-end pipeline. It means the CNN model used in this design can be trained end-to-end. Both ROI(s) detection, as well as the keypoint detection parts, are trained in a single-step process. As illustrated in Fig 5, the complete flow has two parts each with its respective objectives. The first part is performing the hand ROI(s) detection and the next part is performing the keypoints detection. For the ROI proposal generation, the region proposal network uses the feature map F to predict the location of the hand in the image. It is compared with encoded bounding-box information generated using pre-defined anchors and the ground-truth bounding-box coordinates. Using the predicted and the actual bounding box coordinate a regression loss value is calculated, shown as *lossA* in Fig. 5. To train the same model for keypoints detection, the feature map F is masked using the ROI proposal. The resulting feature map F' is used as input to keypoints detection network. The predicted keypoints and the encoded form of ground-truth keypoints are used to calculate a separate loss value referred to as *lossB*. By the iterative minimization of both the loss function *lossA* and *lossB*, the model's parameters are optimized to meet the aforementioned objectives. The details of both the loss functions are as follows

$$loss_{total} = loss_A + loss_B \tag{9}$$

where,

$$loss_A = L_{smooth} + L_{focal}, \tag{10}$$

and

$$loss_B = \sum \hat{Q}.(\hat{z} - z)^2 + \sum (1 - \hat{Q}).(\hat{z} - z)^2. \tag{11}$$

Here, $\hat{z}$ and z are the predicted and the ground-truth values, respectively. L_{smooth} given in (10) is the smooth loss given by

$$L_{smooth} = \begin{cases} |t|, & \text{if } |t| > \beta \\ \dfrac{1}{|\beta|}.t^2, & \text{if } |t| \leq \beta, \end{cases} \tag{12}$$

where, $t = z - \hat{z}$ and β is a threshold at which the loss changes between L_1 and L_2 loss. Its value was 0.5 in our experiment.

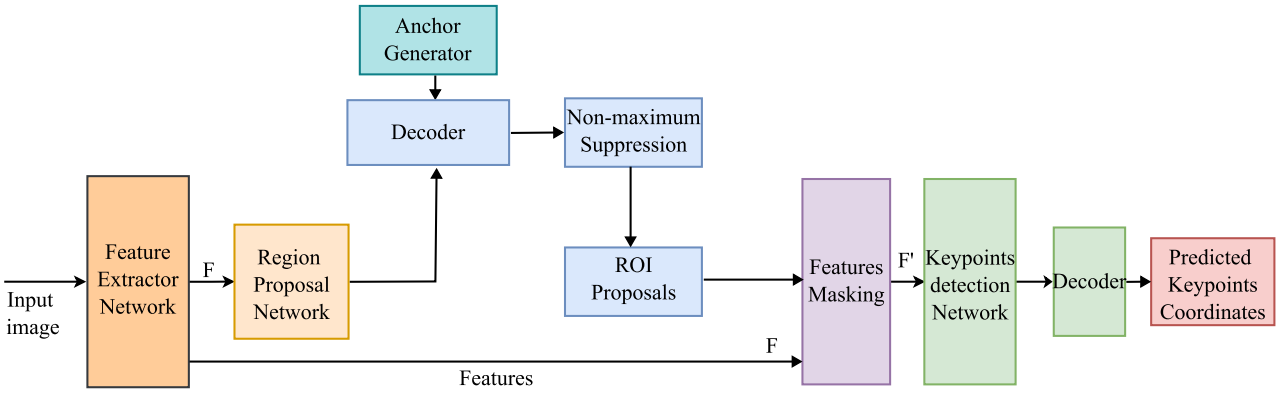


FIGURE 6. The flow diagram for inference pipeline. The flow is slightly different from the training pipeline, especially for the RPN network.

Next, L_{focal} is the focal loss given by

$$L_{focal} = \begin{cases} -\alpha(1-p)^{\gamma}\log(p), & \text{if } z = 1 \\ -(1-\alpha)p^{\gamma}\log(1-p), & \text{otherwise.} \end{cases} \tag{13}$$

where, $\alpha \in [0, 1]$ is a weighting factor. γ is a focusing parameter and p is the model's prediction probability. z is the same as per definition in (11). The values of α and γ were 0.25 and 2, respectively.

The value of $\hat{Q}$ in (11) is calculated as

$$\hat{Q} = \frac{\hat{y}}{\hat{y} + \epsilon}, \tag{14}$$

where, ϵ is a very small value ($= 10^{-6}$) to avoid division by zero.

The SGD optimizer is used to optimize the parameters of the model. The learning rate scheduling approach defined in [47] is used. The initial learning value is set to 10^{-2}. Additionally, to optimize the hyperparameters to the model the methods in [48] and [49] have been referred. Moreover, the model is trained for 150 epochs with a batch size of 16 samples per iteration on an Nvidia V100 GPU machine. The complete architecture is designed using Tensorflow version 2.4 [50].

E. INFERENCE PIPELINE

The flow diagram present in Fig. 6 shows the process followed at the time of inference. The model takes a single RGB image of full size as its input. It first applies a feature extractor layer to the image. The resulting feature map, **F**, is then fed to the RPN layer, which detects all the hands in the image using anchor boxes. The RPN layer computes the offset from an anchor box to a hand-bounding box. These offsets are applied to the anchor boxes followed by a non-maximum suppression (NMS) algorithm. Next, the features map, **F**, is masked using the features-masking technique described in Section III-B. Additionally, as shown in Fig. 1, the keypoint detection process for each detected hand is handled by a separate sub-process in a multi-core CPU system. This helps in running the keypoint detection algorithm for all detected hands in parallel and thus reduces the overall inference time.

IV. EXPERIMENTAL DETAILS

A. DATASETS

The deep learning models are generally data-driven. To make a model perform the hand pose estimation, it needs to be trained with properly annotated datasets. So, we used the OneHand10K dataset [8] to train and validate the proposed hand pose estimation method described in Section III-C. The OneHand10K is a dataset for 2D hand pose estimation. It comprises 10,000 RGB images for training and 1,703 images for testing obtained from online sources. This dataset comprises images with various illumination conditions, different backgrounds, multiple subjects, etc. However, only a single hand is present in every image. This dataset also provides 2D pixel coordinates of 21 keypoints. However, only visible hand keypoints are annotated, and those hidden from view are not annotated.

To train and validate the multiple-hand pose estimation methodology, we used two datasets namely Rendered Handpose Dataset (RHD) [1], and InterHand2.6M [51]. The RHD dataset has 41,258 training and 2,728 testing samples. It provides both RGB images and depth maps of 320 × 320 resolution. It also provides annotations for 21 keypoints. Both single and double hands RGB images are available in this dataset. However, all the images in the dataset are synthetic. The InterHand2.6M dataset is another dataset provided for 3D hand pose estimation of interacting hands. It has two versions, which are categorized by the number of images in the dataset. The smallest among them has a total of 1,275,786 RGB images which are divided into three parts; 738,602 training, 184,287 validation, and 352,897 testing. This dataset provides annotation for 21 keypoints of a hand in the world coordinate system. The annotations are both by humans (H) and machines (M). The images in this dataset are of single, double, interacting, and non-interacting hands. From the training and test set of this dataset, we selected

TABLE 1. The performance of multiple state-of-the-art models as feature extractors in terms of PCK at $\sigma = 0.2$ and mean PCK.

Models	Parameters	FLOPs (M)	Time (ms)	OneHand10K		RHD		InterHand2.6M	
				PCK	mean	PCK	mean	PCK	mean
ResNet50	18.76M	38.359	23.33	0.7221	0.7498	0.8961	0.8847	0.9267	0.9101
DenseNet121	16.15M	19.079	21.89	0.7539	0.7612	0.9252	0.9136	0.9319	0.9261
DenseNet169	23.23M	26.505	28.35	0.7851	0.7975	0.9631	0.9416	0.9791	0.9312
EfficientNetB0	20.45M	17.609	20.34	0.7101	0.7333	0.9035	0.9296	0.9312	0.9483
EfficientNetB1	22.41M	19.257	22.67	0.7647	0.7621	0.9110	0.9023	0.9471	0.9514
EfficientNetB2	29.50M	21.806	24.93	0.7956	0.7711	0.9565	0.9373	0.9498	0.9567

only those images for which *'hand_type_valid'* variable in the annotation is equal to one.

It should be noted that none of these datasets have annotated bounding box coordinates of a hand. We required bounding box coordinates for hand ROI(s) detection. So, the bounding box coordinates are calculated using the 2D coordinates of visible hand keypoints. The top left (p_{lt}) and bottom right (p_{rb}) corner of bounding box is calculated for 2D hand keypoints $\{p_n\}_{n\in K}$ as

$$p_{lt} = \min_{n\in K} p_n, \tag{15}$$

$$p_{rb} = \max_{n\in K} p_n. \tag{16}$$

Additionally, data augmentation has been used to introduce diversity and increase the size of training data. Data augmentation techniques like random flipping, scaling, rotation, cropping, and adding noise are part of the training process. Moreover, variations in hue, saturation, and brightness are also applied to augment human skin color without affecting the texture or hand shape information.

B. EVALUATION PARAMETERS

The quantitative analysis of hand pose estimation is performed using *Probability of Correct Keypoint* (PCK) [52] metric. The PCK value represents a probability that a predicted keypoint lies within a distance threshold σ from its ground-truth position. The PCK value for the *n-th* hand keypoint on the $\mathbb{L}$ samples is denoted by

$$PCK_{\sigma}^{n} = \frac{1}{\mathbb{L}} \sum_{\mathbb{L}} \chi \left(\frac{\|f_n - g_n\|_2}{\max(w_b, h_b)} \right) \tag{17}$$

Here, f_n and g_n are predicted and ground-truth keypoint positions, respectively. The Euclidean distance between f_n and g_n has been normalized by the edge of the bounding box with the maximum length. The bounding box width and height have been denoted by w_b and h_b, respectively. $\chi(t)$ is the indicator function which is defined as

$$\chi(t) = \begin{cases} 1, & \text{if } t \leq \sigma \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Here, σ is a threshold value. We report the results in terms of PCK values at $\sigma = 0.2$ denoted as $PCK_{0.2}$ and averaged mean PCK values.

C. KEYPOINT THRESHOLD

As described in Section III-C2, a threshold $\delta_q \in [0, 1]$ is used to extract the keypoints location from the output features map. To estimate the optimum value of δ_q histogram of all individual keypoints on the validation samples is calculated. And the average is taken across all the samples. The average threshold curve for five keypoints namely the index, the middle, the ring, the thumb, and the little fingertips is shown in Fig. 8. The higher average density is observed approximately after the $\delta_q = 0.5$. Hence, this value is selected to decode the keypoints location from the feature maps.

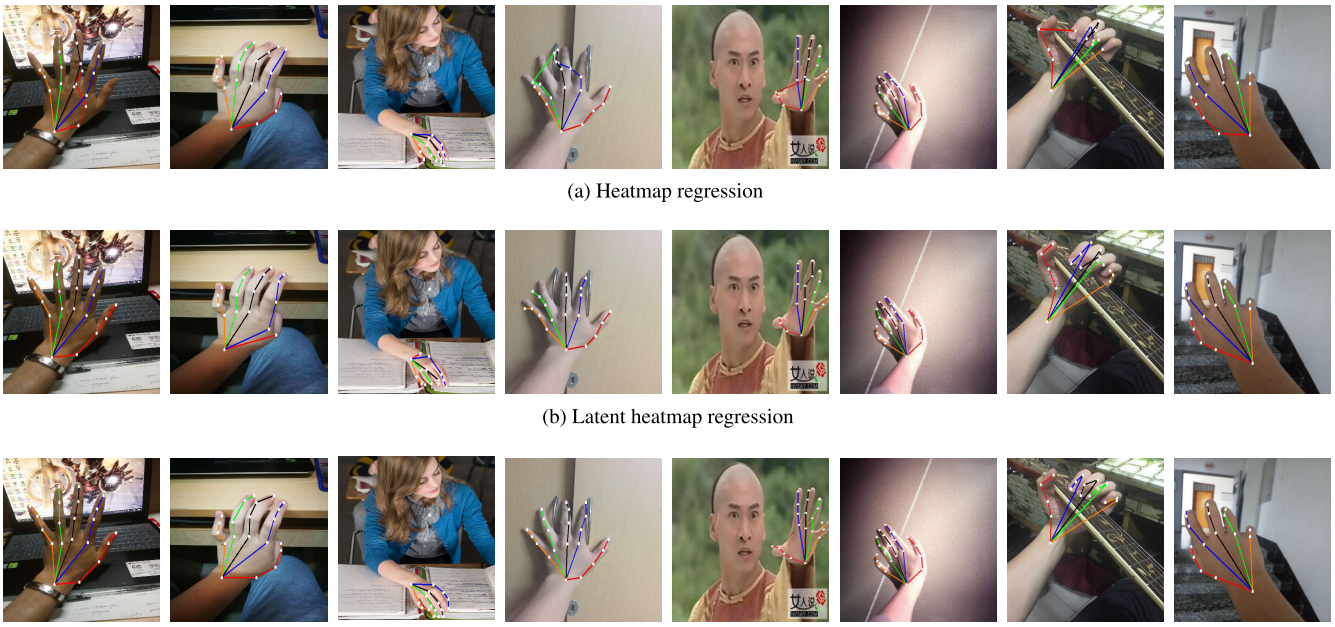
V. RESULTS AND DISCUSSION

A. THE BOTTOM-UP PATHWAY ARCHITECTURE

A few popularly used DL architectures are used as the main feature extractor (the bottom-up pathway shown in Fig. 2). The performance of different architectures viz. ResNet, DenseNet, and EfficientNet on validation sets taken from three different datasets are presented in Table 1. The EfficientB2 architecture helps in achieving the best performance accuracy on the validation set. However, in terms of the number of total model parameters, the EfficientB2 has approximately 29.5 M parameters which is the highest among all the other models used in the proposed architecture. In terms of accuracy, both DenseNet169 and EfficientNetB2 as backbone model has almost the same performance. However, the DenseNet169 performance is slightly better on RHD and InterHand2.6M datasets which have multiple hands images in the validation set. The average inference speed achieved with EfficientNetB2 is 24.93*ms* whereas, with DenseNet169 the value was 28.35 *ms*. Therefore, a tradeoff between inference speed and accuracy has been observed between these two models. Hence, DenseNet169 will be preferred over EfficientNetB2 when the requirement of a particular application is accuracy over inference speed and vice-versa.

B. VALIDATION OF NEAREST-NEIGHBOR BASED POSE ESTIMATION METHOD

A method of 2D hand pose estimation described in Section III-C is compared with commonly used heatmap regression [8], [9], [13], [53] and latent heatmap regression [21], [46] methods used for pose estimation. Both of



(a) Heatmap regression

(b) Latent heatmap regression

(c) Nearest Neighbor (proposed)

FIGURE 7. Visible comparison of proposed hand pose estimation (nearest-neighbor) on samples from the OneHand10k dataset [8] with heatmap-regression and latent heatmap-regression methods.

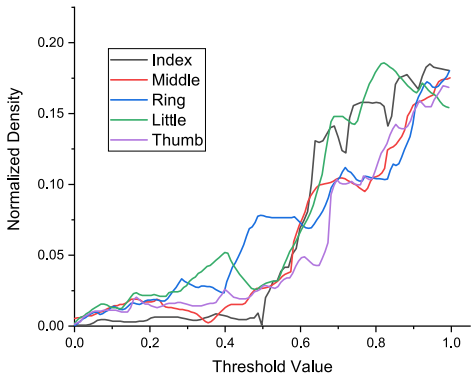


FIGURE 8. The distribution of normalized feature map intensity at different threshold values.

these along with the proposed method were integrated into the proposed deep learning model and the results are compared. The qualitative comparison results are shown in Fig. 7. The skeleton of each finger formed by joining the respective joints is shown in different colors. The red, blue, black, green, and orange colors are used for the thumb, index, middle, ring, and little fingers, respectively. The heatmap regression or latent heatmap regression uses the Gaussian function to encode the keypoints and at the time of inference, the point with maximum probability is taken as the predicted keypoint. A heatmap has a lower resolution than the input image due to downsampling in a DNN. This causes an inevitable quantization error which decreases the position estimation accuracy of a hand joint [54], [55]. The results reported in Table 2 affirm the aforementioned statement. Additionally, the heatmap regression method sometimes leads to incorrect

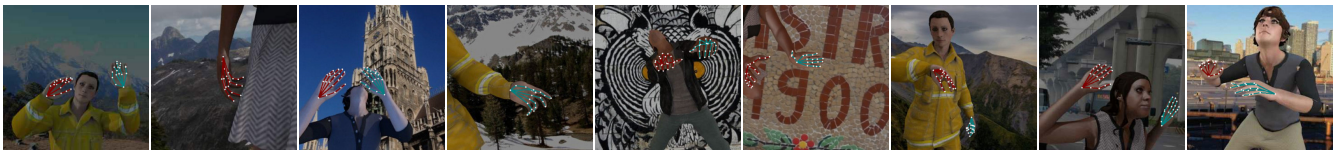
TABLE 2. Performance comparison of the proposed nearest-neighbor method for single-hand pose estimation.

Models	OneHand10K [8]	
	$PCK_{0.2}$	mean
Mask-Pose [8]	0.4473	0.6692
SRHandNet [9]	0.5415	0.7286
OpenPose [13]	0.5992	0.7446
A2J [21]	0.5748	0.5408
Iqbal *et al.* [46]	0.6847	0.7543
SpatialNet [53]	0.5385	0.7046
MMPose [56]	0.7393	0.7443
Hampali *et al.* [57]	0.7084	0.7146
Proposed	0.7851	0.7975

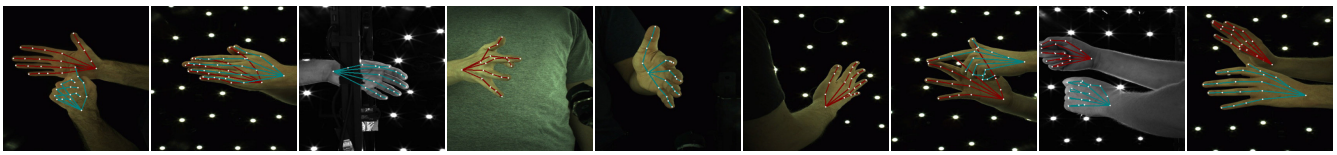
TABLE 3. Comparison of different regression techniques for multi-hand pose estimation.

Method	RHD		InterHand2.6M (H + M)	
	$PCK_{0.2}$	mean	$PCK_{0.2}$	mean
Heatmap regression	0.8547	0.8667	0.8716	0.8754
Latent heatmap regression	0.8838	0.8754	0.8891	0.8914
Nearest-Neighbor (proposed)	0.9631	0.9416	0.9791	0.9313

estimation of keypoints positions or false positives due to a different spatial location having a higher predicted probability as compared to the actual position as evident in Fig. 7a. As latent heatmap regression methodology uses heatmap regression internally similar downside is present in it too. Futhermore, most of the methods based on heatmaps regression use multiple stages of predictor [1], [8], [13] to obtain a refined estimate of joint position. Consequently, it affects the overall speed and slows down the hand joint position estimation process.



(a) RHD [1]

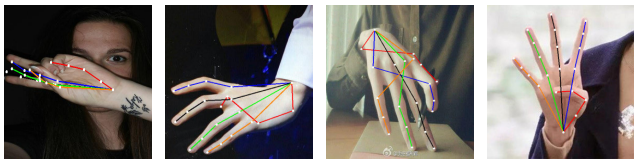


(b) Interhand2.6M [51]

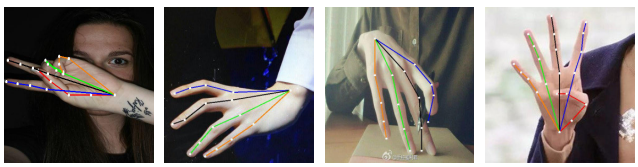
FIGURE 9. Sample results of multiple-hand pose estimation on multiple datasets with the proposed nearest-neighbor HPE method.

TABLE 4. Comparative analysis of multi-hand 2D pose estimation method.

Method	RHD		InterHand2.6M (H + M)	
	$PCK_{0.2}$	mean	$PCK_{0.2}$	mean
Mask-Pose [8]	0.9118	0.8975	0.7984	0.8607
SRHandNet [9]	0.9346	0.9186	0.9310	0.8932
OpenPose [13]	0.6542	0.7588	08303	0.8602
Iqbal *et al*. [46]	0.9282	0.9046	0.9482	0.8892
SpatialNet [53]	0.8287	0.9002	0.6788	0.8148
MMPose [56]	0.9317	0.9123	0.9303	0.8645
Hampali *et al*. [57]	0.8742	0.8367	0.8946	0.8662
Santavas [58]	0.6725	0.8004	0.9288	0.8865
Guo *et al*. [59]	0.8698	0.8994	0.9118	0.8845
Li *et al*. [60]	0.8602	0.8225	0.9546	0.9049
Nearest Neighbor (Proposed)	0.9631	0.9416	0.9791	0.9313



(a) MMPose



(b) Proposed method

FIGURE 10. A few test results of MMPose and proposed method for 2D pose estimation on OneHand10k dataset samples.

C. VALIDATION OF MULTIPLE-HAND POSE ESTIMATION METHOD

The design of the proposed methodology facilitates the integration of different pose estimation techniques like heatmap regression or latent heatmap regression in the architecture. A comparative study was conducted to identify the best-performing technique suitable for multi-hand pose estimation. Comparative results on the test set from two different datasets are reported in Table 3. It can be inferred from the reported values that the proposed nearest-neighbor-based technique has an edge over the other methods in terms of performance. The same can also be validated by doing a visible analysis. A few sample results from two different datasets using the proposed method are shown in Fig. 9. The sample images shown present both hands' keypoints detection results. Red and cyan-colored skeletons with white dots are used to present the keypoints detection results of the right and left hand, respectively. The results demonstrate that the keypoints' positions are estimated with great precision using the proposed method.

A quantitative comparison is also conducted with different recently proposed methods for 2D hand pose estimation. The results are reported in Table 4. The methods used in comparison are based on linear regression [58], heatmap regression [8], [9], [13], [56], [57], latent heatmap regression [46], [60], and graph-based method [59]. A graph-based technique to proposed by Guo et al. [59] and its performance is competitive. However, the major limitation of the graph-based approach is that it works only for a fixed number of points, and in its current form, it is unsuitable for real-time applications. The MMPose [56] methods are based on a top-down pose estimation approach which first uses a detector for hand localization followed by a heatmap regression-based pose estimation approach. The MMPose model has comparatively subpar performance when tested on OneHand10K and RHD datasets. A few comparative sample results are presented in Fig. 10. There are a few miss-classified keypoints. Moreover, the MMPose method was able to detect keypoints from a single hand only. The results of multiple hand image samples were weak. Vision transformer based proposed in [57] just uses a single channel heatmap image instead of multiple channels used commonly (for heatmap regression) to get the 2D location of hand keypoints. It uses a Gaussian function with a fixed standard deviation to encode the keypoints location. It would be quite difficult to separate one keypoints from another if two or more keypoints are very close to each other in a hand pose or if the area occupied by the hand is small in the image.

FIGURE 11. Inference on the small hands.

This affects its performance on the full-size image and needs to rely on a top-down approach to segment the hand region first before pose estimation. Unlike, the aforementioned approaches, we have proposed a single-step approach to estimate the 2D hand pose estimation for multiple hands. Thus, overcoming some of the mentioned weaknesses of these methods and performing at par or better than the existing approaches.

To have a fair comparison, all these methods are trained on the same datasets using the parameters defined for optimum performance in their respective papers. Moreover, the existing model and its weight if provided by the authors were used for comparative analysis. In comparison to all the methods aforementioned, the method proposed in this paper is the best performing.

D. SCOPE OF IMPROVEMENT

We have observed that the performance of the proposed keypoints estimation algorithm drops when the hand occupies a small area in the whole image referred to as a small hand case. The small is defined when the ratio of the hand region bounding box width or height to the image width or height is less than 0.3. An example is presented in Fig. 11. In the right image of Fig. 11, there are some false positive keypoints whereas, in the right image, some predicted keypoints are not at the correct location. This drop in performance is due to the fact that at a lower resolution of hand ROI, the effect of self-occlusion is more in comparison to high-resolution hand ROI. Additionally, the similarity of fingers affects the performance at lower resolution. The improvement in the proposed hand pose estimation on lower-resolution images and small hands will be pursued in our future work.

VI. CONCLUSION

In this paper, we propose a methodology for 2D hand pose estimation from a monocular RGB image. The primary focus is to propose a methodology that will facilitate multiple-hand pose estimation. Through a latent hand(s) ROI detection technique and feature sharing, an end-to-end pipeline is proposed using which the primary objective is met. The same has been validated by performing experiments on publically available datasets that provide annotated hand keypoints positions for both hands of a subject. However, obtaining datasets with more than one subject and properly annotated hand keypoints is difficult. Along with the multiple-hand pose estimation algorithm, a new technique using nearest-neighbor grid cells is introduced in this paper. The proposed method has improved the accuracy of hand keypoint detection and it is effective in locating keypoint positions for several hand postures. A combination of multiple hands pose estimation technique and nearest-neighbor grid cells methodology work for multiple hand poses and scales of the hand. In terms of performance metric, the $PCK_{0.2}$ values obtained are above 95% on Interhand2.6M and RHD datasets used for validation indicating the reliable performance of the proposed algorithms. However, there are chances that performance will degrade if the scale of the hand in the input image is very small. Nevertheless, using parallel processing a reliable inference time is achievable allowing the algorithm to be used for consumer applications such as virtual typing, airwriting, etc. In this work, only 2D hand pose estimation is presented, however, a new way to estimate 3D hand pose estimation from RGB images will also be explored.

ACKNOWLEDGMENT

The support and the resources provided by PARAM Shivay Facility under the National Supercomputing Mission, Government of India at the Indian Institute of Technology (BHU), Varanasi is gratefully acknowledged.

REFERENCES

[1] C. Zimmermann and T. Brox, "Learning to estimate 3D hand pose from single RGB images," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4913–4921.

[2] G. Modanwal and K. Sarawadekar, "Towards hand gesture based writing support system for blinds," *Pattern Recognit.*, vol. 57, pp. 50–60, Sep. 2016.

[3] J.-H. Song, K. Kong, and S.-J. Kang, "Dynamic hand gesture recognition using improved spatio-temporal graph convolutional network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 6227–6239, Sep. 2022.

[4] M. Wang, C. Luo, B. Ni, J. Yuan, J. Wang, and S. Yan, "First-person daily activity recognition with manipulated object proposals and non-linear feature fusion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 2946–2955, Oct. 2018.

[5] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Comput. Vis. Image Understand.*, vol. 108, nos. 1–2, pp. 52–73, Oct. 2007.

[6] B. Doosti, "Hand pose estimation: A survey," 2019, *arXiv:1903.01013*.

[7] M. M. Alam, M. T. Islam, and S. M. M. Rahman, "Unified learning approach for egocentric hand gesture recognition and fingertip detection," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108200.

[8] Y. Wang, C. Peng, and Y. Liu, "Mask-pose cascaded CNN for 2D hand pose estimation from single color image," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 11, pp. 3258–3268, Nov. 2019.

[9] Y. Wang, B. Zhang, and C. Peng, "SRHandNet: Real-time 2D hand pose estimation with simultaneous region localization," *IEEE Trans. Image Process.*, vol. 29, pp. 2977–2986, 2020.

[10] S. Gattupalli, A. R. Babu, J. R. Brady, F. Makedon, and V. Athitsos, "Towards deep learning based hand keypoints detection for rapid sequential movements from RGB images," in *Proc. 11th Pervasive Technol. Rel. Assistive Environments Conf.*, Jun. 2018, pp. 31–37.

[11] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, Jan. 2021.

[12] J. Lauer, M. Zhou, S. Ye, W. Menegas, S. Schneider, T. Nath, M. M. Rahman, V. Di Santo, D. Soberanes, G. Feng, V. N. Murthy, G. Lauder, C. Dulac, M. W. Mathis, and A. Mathis, "Multi-animal pose estimation, identification and tracking with DeepLabCut," *Nature Methods*, vol. 19, no. 4, pp. 496–504, Apr. 2022.
[13] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4645–4653.
[14] P. Panteleris, I. Oikonomidis, and A. Argyros, "Using a single RGB frame for real time 3D hand pose estimation in the wild," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2018, pp. 436–445, doi: 10.1109/WACV.2018.00054.
[15] D. J. Sturman and D. Zeltzer, "A survey of glove-based input," *IEEE Comput. Graph. Appl.*, vol. 14, no. 1, pp. 30–39, Jan. 1994.
[16] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn, "FingARtips: Gesture based direct manipulation in augmented reality," in *Proc. 2nd Int. Conf. Comput. Graph. Interact. Techn. Australasia South East Asia*, Jun. 2004, pp. 212–221.
[17] K. Dorfmuller-Ulhaas and D. Schmalstieg, "Finger tracking for interaction in augmented environments," in *Proc. IEEE ACM Int. Symp. Augmented Reality*, Oct. 2001, pp. 55–64.
[18] J. Sanchez-Riera, K. Srinivasan, K.-L. Hua, W.-H. Cheng, M. A. Hossain, and M. F. Alhamid, "Robust RGB-D hand tracking using deep learning priors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 9, pp. 2289–2301, Sep. 2018.
[19] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 824–832.
[20] C. Wan, A. Yao, and L. Van Gool, "Hand pose estimation from local surface normals," in *Computer Vision—ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 554–569.
[21] F. Xiong, B. Zhang, Y. Xiao, Z. Cao, T. Yu, J. T. Zhou, and J. Yuan, "A2J: Anchor-to-joint regression network for 3D articulated pose estimation from a single depth image," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 793–802.
[22] W. Wu, C. Li, Z. Cheng, X. Zhang, and L. Jin, "YOLSE: Egocentric fingertip detection from single RGB images," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 623–630.
[23] Y. Li, X. Wang, W. Liu, and B. Feng, "Pose anchor: A single-stage hand keypoint detection network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 2104–2113, Jul. 2020.
[24] V. Athitsos and S. Sclaroff, "Estimating 3D hand pose from a cluttered image," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 2003, pp. II–432.
[25] G. Rogez, M. Khademi, J. S. Supančič III, J. M. M. Montiel, and D. Ramanan, "3D hand pose detection in egocentric RGB-D images," in *Computer Vision—ECCV 2014 Workshops*. Cham, Switzerland: Springer, 2015, pp. 356–371.
[26] M. Baydoun, A. Betancourt, P. Morerio, L. Marcenaro, M. Rauterberg, and C. Regazzoni, "Hand pose recognition in first person vision through graph spectral analysis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 1872–1876.
[27] T. Grzejszczak, M. Kawulok, and A. Galuszka, "Hand landmarks detection and localization in color images," *Multimedia Tools Appl.*, vol. 75, no. 23, pp. 16363–16387, Dec. 2016.
[28] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4724–4732.
[29] D. Kong, H. Ma, Y. Chen, and X. Xie, "Rotation-invariant mixed graphical model network for 2D hand pose estimation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2020, pp. 1535–1544, doi: 10.1109/WACV45572.2020.9093638.
[30] D. Gupta, B. Artacho, and A. Savakis, "HandyPose: Multi-level framework for hand pose estimation," *Pattern Recognit.*, vol. 128, Aug. 2022, Art. no. 108674.
[31] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proc. ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 483–499.
[32] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 472–487.
[33] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5686–5696.
[34] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2353–2362.
[35] S. Kreiss, L. Bertoni, and A. Alahi, "PifPaf: Composite fields for human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11969–11978.
[36] D. Shi, X. Wei, L. Li, Y. Ren, and W. Tan, "End-to-end multi-person pose estimation with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11059–11068.
[37] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019, *arXiv:1904.07850*.
[38] W. Mao, Z. Tian, X. Wang, and C. Shen, "FCPose: Fully convolutional multi-person pose estimation with dynamic instance-aware convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9030–9039.
[39] D. Shi, X. Wei, X. Yu, W. Tan, Y. Ren, and S. Pu, "InsPose: Instance-aware networks for single-stage multi-person pose estimation," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 3079–3087, doi: 10.1145/3474085.3475447.
[40] X. Nie, J. Feng, J. Zhang, and S. Yan, "Single-stage multi-person pose machines," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6950–6959.
[41] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944.
[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[43] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[44] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, K. Chaudhuri and R. Salakhutdinov, Eds., 2019, pp. 6105–6114.
[45] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, Aug. 2006, pp. 850–855.
[46] U. Iqbal, P. Molchanov, T. Breuel, J. Gall, and J. Kautz, "Hand pose estimation via latent 2.5D heatmap regression," in *Computer Vision—ECCV 2018*. Cham, Switzerland: Springer, 2018, pp. 125–143.
[47] P. Mishra and K. Sarawadekar, "Polynomial learning rate policy with warm restart for deep neural network," in *Proc. TENCON - IEEE Region 10 Conf. (TENCON)*, Oct. 2019, pp. 2087–2092.
[48] D. Singh, M. Kaur, J. M. Alanazi, A. A. AlZubi, and H.-N. Lee, "Efficient evolving deep ensemble medical image captioning network," *IEEE J. Biomed. Health Informat.*, vol. 27, no. 2, pp. 1016–1025, Feb. 2023.
[49] D. Singh, M. Kaur, M. Yaseen Jabarulla, V. Kumar, and H.-N. Lee, "Evolving fusion-based visibility restoration model for hazy remote sensing images using dynamic differential evolution," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–14, 2022.
[50] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, *arXiv:1603.04467*.
[51] G. Moon, S.-I. Yu, H. Wen, T. Shiratori, and K. M. Lee, "InterHand2.6M: A dataset and baseline for 3D Interacting hand pose estimation from a single RGB image," in *Computer Vision—ECCV 2020*. Cham, Switzerland: Springer, 2020, pp. 548–564.
[52] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2878–2890, Dec. 2013.
[53] T. Pfister, J. Charles, and A. Zisserman, "Flowing ConvNets for human pose estimation in videos," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1913–1921.
[54] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 529–545.
[55] B. Yu and D. Tao, "Heatmap regression via randomized rounding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8276–8289, Nov. 2022.
[56] MMPose Contributors. (2020). *Openmmlab Pose Estimation Toolbox and Benchmark*. [Online]. Available: https://github.com/open-mmlab/mmpose
[57] S. Hampali, S. D. Sarkar, M. Rad, and V. Lepetit, "Keypoint transformer: Solving joint identification in challenging hands and object interactions for accurate 3D pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11080–11090.

[58] N. Santavas, I. Kansizoglou, L. Bampis, E. Karakasis, and A. Gasteratos, "Attention! A lightweight 2D hand pose estimation approach," *IEEE Sensors J.*, vol. 21, no. 10, pp. 11488–11496, May 2021.
[59] S. Guo, E. Rigall, L. Qi, X. Dong, H. Li, and J. Dong, "Graph-based CNNs with self-supervised module for 3D hand pose estimation from monocular RGB," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 4, pp. 1514–1525, Apr. 2021.
[60] M. Li, J. Wang, and N. Sang, "Latent distribution-based 3D hand pose estimation from monocular RGB images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4883–4894, Dec. 2021.

PURNENDU MISHRA received the Bachelor of Engineering degree in electronics and communication engineering from the Birla Institute of Technology, Mesra, Ranchi, India, in 2012, and the Master of Technology degree in electronics and communication engineering from the National Institute of Technology, Jalandhar, India, in 2015.

He is currently a Research Scholar with the Department of Electronics Engineering, Indian Institute of Technology (BHU) Varanasi, Varanasi, India. He is supervised by Dr. Kishor Sarawadekar, an Associate Professor with the Department of Electronics Engineering, IIT (BHU) Varanasi. His research interests include image processing, computer vision, and deep learning. He has published papers in reputed journals and conferences in these domains. He is also involved on developing novel deep-learning models for medical image analysis and segmentation. He is passionate about working in the technical domain, especially artificial intelligence.

KISHOR SARAWADEKAR (Senior Member, IEEE) received the M.Tech. and Ph.D. degrees in microelectronics and VLSI design from Indian Institute of Technology, Kharagpur, India, in 2007 and 2011, respectively.

He was a Software Engineer with Xilinx India Technology Services Pvt., Ltd., Hyderabad, India, where he was involved in creating reference designs on the FPGA platform. He is currently an Associate Professor with the Department of Electronics Engineering, Indian Institute of Technology (IIT-BHU) Varanasi, Varanasi. He has about 20 years of industry and academic experience. He was a Nodal Officer with Visveswaraya Ph.D. Scheme for Electronics and IT, Ministry of Electronics and Information Technology, Government of India. Three research scholars have earned their Ph.D. under his supervision and 26 students have completed their master's thesis. His research interests include algorithms and architectures for image/video processing, and image coding systems, such as HEVC, JPEG 2000, bio-medical image processing, and related low-power VLSI designs. Prior to this, he had brief work experience with the Defence Institute of Technology, Pune.

Dr. Sarawadekar was a recipient of the Visvesvaraya Young Faculty Fellowship. He has been a Reviewer of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON IMAGE PROCESSING, *Journal of Real-Time Image Processing*, *Defence Science Journal*, DRDO, India, and *Journal of Electrical and Computer Engineering*. He was the Chair of the VLSI Design and Automation Track in the IEEE Technical symposium (Techsym) 2011, the Digital Integrated Circuits-2 Track in 2016 IEEE 59th Midwest Symposium on Circuits and Systems (MWSCAS), Abu Dhabi, United Arab Emirates, and the Diagnostic and Imaging Technologies Session in the BIT's Annual World Congress of Regenerative Medicine and Stem Cell-2017 (RSMC), Singapore.