Dynamic Iterative Refinement for Efficient 3D Hand Pose Estimation

John Yang[1]*, Yash Bhalgat[2], Simyung Chang[3], Fatih Porikli[2], Nojun Kwak[1]†

[1]Seoul National University, Seoul, Korea
[2]Qualcomm AI Research, Qualcomm Technologies, Inc., San Diego, CA, US‡
[3]Qualcomm AI Research, Qualcomm Korea YH, Seoul, Korea
{yjohn, nojunk}@snu.ac.kr, {ybhalgat, simychan, fporikli}@qti.qualcomm.com

Abstract

While hand pose estimation is a critical component of most interactive extended reality and gesture recognition systems, contemporary approaches are not optimized for computational and memory efficiency. In this paper, we propose a tiny deep neural network of which partial layers are recursively exploited for refining its previous estimations. During its iterative refinements, we employ learned gating criteria to decide whether to exit from the weight-sharing loop, allowing per-sample adaptation in our model. Our network is trained to be aware of the uncertainty in its current predictions to efficiently gate at each iteration, estimating variances after each loop for its keypoint estimates. Additionally, we investigate the effectiveness of end-to-end and progressive training protocols for our recursive structure on maximizing the model capacity. With the proposed setting, our method consistently outperforms state-of-the-art 2D/3D hand pose estimation approaches in terms of both accuracy and efficiency for widely used benchmarks.

1. Introduction

Hand pose estimation (HPE) is an essential task for augmented reality and virtual reality (collectively called as "extended reality (XR)") systems. For instance, to enable hand-based interactions with objects in XR environments, accurate real-time estimates of the positions of hand joints in 3D world coordinates are needed. Since hand gestures reflect elementary human behavioral patterns, hand pose tracking enables several downstream AI applications such as gesture recognition [22, 34, 41] and human-computer interactions [7, 20]. Although many state-of-the-art HPE approaches [5, 6, 28, 45, 49, 55] achieve high accuracy, they rely on large and complex model architectures, which incur a substantial computational cost. Therefore, such models are often unsuitable for relatively low-power computing systems, like wearables or hand-held XR devices [1, 8, 35].

To meet the constraints of resource-limited devices, [17, 12, 46, 47] have proposed neural network architectures with dynamic inference graphs that are conditional on the input. Recently, [9] introduced an analogous approach for 3D HPE that exploits a Gaussian-kernel-based gating mechanism to adaptively combine the predictions of coarse and fine pose encoders, thus achieving a reduction in the GFLOPs for inference. While it alleviates the run-time complexity, the memory size required to deploy both pose encoders makes the method still infeasible for memory-constrained real-world applications. In this work, we tackle both the run-time efficiency and memory usage challenges of 3D HPE with a modular network whose capacity can be dynamically amplified through recursive exploitation of the network's parameters.

Recursive usage of layers with shared parameters has been proposed by various works [13, 27, 37, 50] as a tool to match the performance of deeper networks with fewer parameters. Considering the distinctive semantic encoding at each layer in a neural network [51], recursively exploited layers must manage gaps between higher-level features and lower-level ones. Thus, rather than directly re-feeding the output features to the consecutive recursive layers, we employ 'refinements' of original input features using novel attention-augmentation (more details in Sec. 3.1). This allows the recursive component of the deep network to focus on a distinct level of semantic information in each iteration.

With the above points in mind, we present Dynamic Iterative Refinement Network (DIR-Net), a modular weight-sharing network of which the capacity is adaptively amplified through dynamic recursive use of partial layers. DIR-Net is composed of a few components: *Refiner* that handles

*Work done as an intern at Qualcomm Technologies, Inc.
†Corresponding Author
‡Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

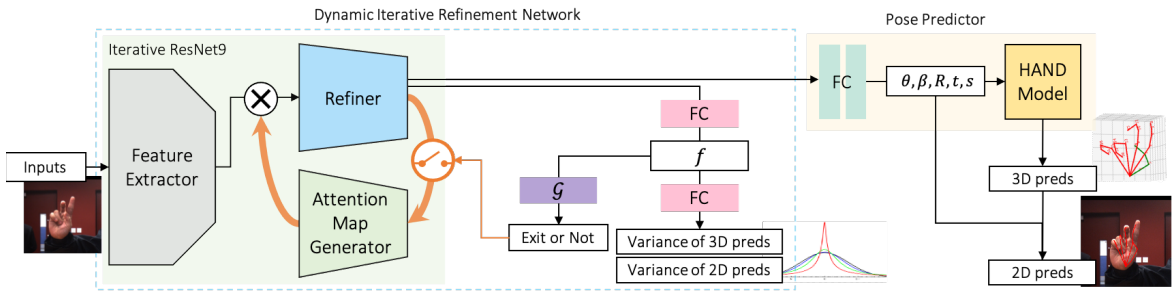


Figure 1: **Dynamic Iterative Refinement Network (DIR-Net).** DIR-Net is composed of an *Iterative ResNet9* (IR9) backbone network, an *Uncertainty Estimation module* and a *Gating function*. The backbone IR9 consists of a *Feature Extractor* (FE), a *Refiner* (RF) and an *Attention Map Generator* (AMG). For an image input, DIR-Net predicts 2D/3D hand joint locations. The model then refines its predictions with a repeated use of RF and AMG. Such re-use of RF + AMG is continued until the gating function tells to exit the loop. DIR-Net's inference is efficient in terms of both computation and memory due to its lightweight design.

distribution shifts in the attention-augmented input feature maps and produces refined predictions, *Attention Map Generator* that refines the input features, and *Gating Function* that dynamically controls the number of iterations. Along with the localized keypoint (joint) estimations, our backbone network is also trained to estimate the uncertainty in its own predictions. At every iteration, the gating function relies on these uncertainty estimations to decide whether the network should refine its current keypoint estimations or exit. This leads to per-sample adaptive inference in our model.

Overall, our main contributions in this work are:

- We introduce a lightweight architecture, DIR-Net, parts of which are utilized recursively while incorporating attention-augmentation and gating for dynamic refinement of the hand pose estimations.
- We propose a variance-based and also a reinforcement learning approach for dynamic gating that directly and indirectly exploits the uncertainty predictions of the estimated 2D/3D pose.
- We investigate the effectiveness of progressive and end-to-end training protocols for the inference-time efficiency of our recursive structure.
- With a comprehensive set of experiments and ablation studies, we show that our method achieves state-of-the-art performance in terms of both accuracy and efficiency on two widely used HPE benchmarks.

2. Related Work

Adaptive Neural Computing. While going deeper in neural networks has helped achieve state-of-the-art performance on several image recognition tasks, early research works [44, 32] have pointed out that the task-specific complexity varies widely across input samples, and only a small fraction of inputs require processing by the entire network. For enabling energy-efficient implementations of deep networks, [17, 29, 31] have proposed selecting different inference paths conditional on the input to optimize the overall trade-off between efficiency and accuracy. Residual networks [14, 15] have been a common choice as the backbone architecture for many dynamic inference approaches either through adaptive block removals [48] or skips [25, 46, 47]. Such dynamic gating turns on-and-off some of the blocks so that a network outputs its predictions through shortened forwarding paths.

Similar to the idea of shortening the inference paths, early-exiting methods adaptively exit from neural networks before the inference reaches its final layer [26, 44]. For classification tasks, multiple classifiers are added along the feed-forward path. There is an early exit mechanism from the main inference path if the confidence of the earlier classifiers satisfies a pre-defined exiting criterion. These methods typically require heuristic values of confidence [26] or entropy [44]. Adaptive Neural Trees [43] perform both dynamic gating and early exiting as its inference path is determined by the decisions at the leaves of a neural tree.

Recurrent architectures for Image Recognition. Recurrent use of network layers with shared parameters has shown superior efficiency in a few recent works. Yoo *et al.* [50] propose an iterative feature map generation method in which feature maps in different resolutions are iteratively generated by recurrently passing a network structured with inverted residual blocks. RNNPool [37] also uses a recursive inference for efficient downsampling of the feature maps. Their method and ours share the notion of effectively bringing the higher-level feature information to the lower-level to overcome the limited semantic information in the lower-level features due to the shallow overall structure. However, their recurrent use of parameters does not work in an adaptive manner, operating with a fixed inference graph for all inputs.

As an early work, adaptive and iterative use of a network is proposed in [12] with the condition of a *ponder* cost. The recurrent inference is adaptively repeated based on computation cost allowance. The later adaptive and iterative inference methods are proposed mainly with blocks of deep residual networks [13, 27]. While the original residual block is recursively exploited in [13], a neural recursive module that is inspired by residual blocks is introduced in [27]. Our work is similar to the works of [13, 27] where parts of a model are recursively iterated to maximize the capacity of the parameters.

Adaptive 3D HPE. 3D hand pose estimation problems have been actively studied during the last decade [5, 19, 30, 49, 55], including pre-deep-learning methods [33, 52]. However, despite its importance, 3D hand pose estimation methods have been rarely considered with adaptive computing. Recently, ACE-Net is proposed for efficient hand tracking [9]. It is largely composed of two modular networks: one shallow, coarse pose encoder and another with a deeper, fine pose encoder. ACE-Net adaptively selects either encoder to feed-forward an input image. While this reduces computational cost during test time, the memory requirement for deploying a network with two separate encoders is infeasible for resource-constrained scenarios.

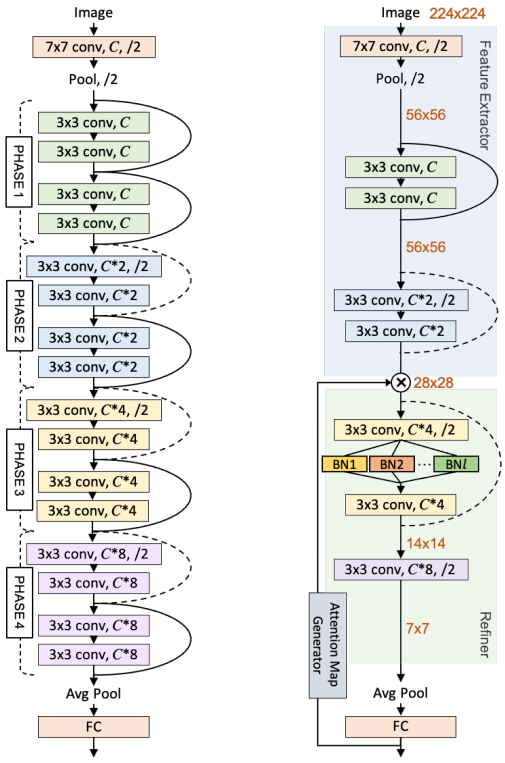
3. Iterative Refinement Network

In this section, we introduce our Iterative Refinement network, a modular weight-sharing neural model with iterative exploitation of parameters that yield refined 3D hand pose estimations in every iteration via attention-augmentation. As illustrated in Figure 1, our proposed overall architecture mainly consists of Feature Extractor (FE), Refiner (RF), Attention Map Generator (AMG), Gating Function ($\mathcal{G}$), an uncertainty estimation module and the 2D/3D hand pose predictor. During inference, a monocular RGB image of a hand is passed through the FE and sequentially to RF. The outputs of RF are then fed to (1) 2D/3D hand pose predictor, (2) uncertainty estimator, and (3) AMG (conditional on the gating function's output).

Note that, in Sec. 4, we describe two possible ways to perform the gating - (1) using a simple heuristic based on the uncertainty outputs, or (2) an RL agent ($\mathcal{G}$) trained using a reward function that optimizes the accuracy-efficiency trade-off. Only the latter case is illustrated in Figure 1.

3.1. Iterative ResNet9 Backbone

Recursive inference allows for the usage of higher-level features to refine lower-level features in order to exploit the capacity of network parameters maximally. Such utilization of the network parameters enables a network with a much smaller size to have similar prediction accuracy as that of a more complex network [13, 27, 37, 53]. Our tiny backbone network, where the default model capacity is low, is amplified with each iteration of recursive use.



(a) ResNet18 (b) Iterative ResNet9

Figure 2: Iterative ResNet9 (IR9) follows a modular structure similar to ResNet18. The intention of the Refiner and Attention Map Generator is to reduce the overall size and complexity of ResNet18 while preserving the task-specific performance. IR9 uses distinct BatchNorm layers to handle distribution shifts in every iteration.

Inspired by ResNet18 [14], we design a modular network, ResNet9, with reasonably low complexity while keeping the downsampling characteristic of its original structure. Such modularization can be made with any network to exploit our proposed recursive refinement approach. We opted for a residual architecture due to its easily separable structure of block units, as done in relevant works [13, 25, 27, 50]. As shown in Figure 2b, the outputs of the ResNet9 backbone are refined by an iterative usage of the last few layers of the network. With such a mechanism, the proposed Iterative ResNet9 (IR9) requires much lesser memory to be deployed in systems due to its overall small model size. The components of this iterative backbone architecture are described in more detail below.

Feature Extractor (FE): This component of the backbone network is responsible for encoding low-level image features. Since Feature Extractor is not operated recursively, its feature encoding involves heavy down-sampling of the feature maps to reduce the cost of the recursive computa-

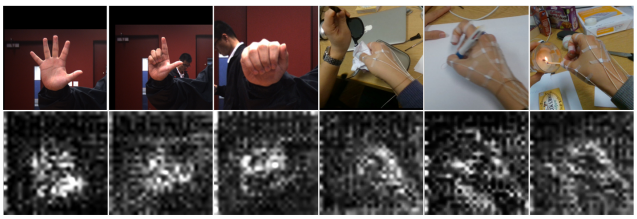


Figure 3: Images in the top row are the inputs and ones in the bottom row are normalized features that are computed with attention maps.

tion that follows. In our best[1] model, FE reduces the spatial dimensions of image inputs from 224×224 to 28×28.

Refiner (RF): This part is recursively "looped" in our proposed architecture. In every loop iteration (except the first), the attention-augmented feature maps are fed to this module, which refines the predictions of the previous iteration. Due to attention-augmentation, the distribution of the input feature map changes in each iteration. Hence, a separate Batch Normalization layer [18] is used in each iteration to handle the statistic shifts in input feature maps that are attention-augmented [13, 27].

Attention Map Generator (AMG): This module has an upsampling decoder architecture that outputs an attention map tensor with values in the range $[0, 1]$ and of the same size as the feature map output of FE. To effectively upscale the outputs of RF without any significant computational cost, the decoder is mainly composed of pixel-shuffle layers [39], which transfer the channel-wise information to the spatial domain with a negligible cost. The main purpose of AMG is to merge information across different depths of the backbone network by spatially re-projecting the higher-level features to the lower-level feature maps. In doing so, the attention map tensor is element-wise multiplied with the output feature maps of FE. This allows dense connectivity that spatially links higher-level features and lower-level 2D feature planes [17, 39]. Figure 3 shows the effect of the attention maps on FE's output feature maps. The images in the bottom row shows the Hadamard product of FE feature output and attention map, averaged over channel axis and normalized.

3.2. Pose Predictor

The structure of our pose predictor mainly follows the works of [4, 49], consisting of two fully connected layers and a MANO hand mesh deformation model [36]. MANO model takes low-dimensional pose and shape embeddings, respectively θ and β, as inputs for controlling the 3D hand mesh outputs: $J(\theta, \beta) = R_\theta(J(\beta))$. The location of joints $J(\beta)$ can be globally rotated based on the pose θ, denoted as R_θ, to obtain a final hand model with corresponding 3D coordinates of 21 joints of a hand.

Our method takes cropped hand images as inputs x. The output of FE (denoted by $\mathcal{F}(x)$) is fed to RF $\mathcal{R}(.)$ along with an attention map $\mathcal{M}^l$ generated at each recursive iteration $l \in \{0, 1, 2, ..., l_{max}\}$. Our pose predictor takes $\mathcal{R}(\mathcal{F}(x))$ as input when $l = 0$ and $\mathcal{R}(\mathcal{F}(x), \mathcal{M}^l)$ as input when $l > 0$, and predicts a rotation matrix $R \in SO(3)$, a translation $t \in \mathbb{R}^2$ and a scaling factor $s \in \mathbb{R}^+$ along with its pose θ and shape β:

$$\theta, \beta, R, t, s = \begin{cases} \Psi_{pose}(\mathcal{R}(\mathcal{F}(x))), & l = 0 \\ \Psi_{pose}(\mathcal{R}(\mathcal{F}(x), \mathcal{M}^l)), & l > 0, \end{cases} \quad (1)$$

where $\Psi_{pose}(\cdot)$ represents a neural network with two fully-connected layers. 3D locations of joints $J(\theta, \beta)$ are obtained from MANO, and the 2D keypoint estimates are obtained by re-projecting these 3D locations to the 2D image plane with a weak-perspective camera model using the estimated camera parameters $\{R, t, s\}$:

$$\begin{aligned} \hat{J}_{3D} &= J(\theta, \beta) \\ \hat{J}_{2D} &= s\Pi R J(\theta, \beta) + t \end{aligned} \quad (2)$$

where Π represents orthographic projections.

With the re-projected 2D joint location estimations, the network can implicitly learn 3D joint locations with 2D labels [4, 49]. To train the pose predictor, we use $L1$ and $L2$ losses for 2D and 3D predictions respectively:

$$\begin{aligned} L_{2D} &= ||J^{gt}_{2D} - \hat{J}_{2D}||_1, \\ L_{3D} &= ||J^{gt}_{3D} - \hat{J}_{3D}||_2. \end{aligned} \quad (3)$$

Since the error for 2D estimations is calculated at pixel-level in image planes, L_{2D} is desired to be more robust. The combination of using L_1-norm for L_{2D} and L_2-norm for L_{3D} has been found to be the best [4, 11, 16]. We do not use the vertices of MANO hand mesh for training because their ground truths are not available during experiments. For the final loss, we also include objective terms that regularize pose and shape parameters, as done in [4].

4. Dynamic Exiting Mechanisms

Although a *maximum* number of loop iterations is set in our Iterative Refinement network, not all input images require that many iterations of refinement. We propose alternative gating policies to determine when to stop iterating for each sample. The resulting architecture is called Dynamic Iterative Refinement Network (DIR-Net). Overall, two gating policies are proposed: (1) based on heuristic variance thresholds and (2) decisions of a reinforcement learning agent.

[1] In Sec. 6.2, we provide a detailed ablation study of other downsampling ratios and their respective computation vs accuracy trade-offs.

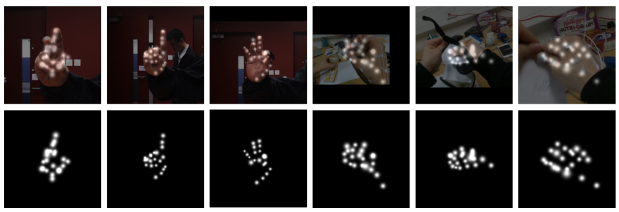


Figure 4: Heatmaps generated based on uncertainty for its 2D joint localization estimations.

4.1. Uncertainty Based Exiting

To decide whether to proceed to a next loop of recursive inference of the model, the model should be aware how certain its current predictions are. To this end, the model estimates variances for its 2D/3D keypoint predictions, by estimating a probability distribution instead of only joint locations. We simplify the problem and assume coordinates of joints are independent so we can use univariate Gaussians:

$$P_W(J) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(J-\hat{J})^2}{2\sigma^2}} \tag{4}$$

where W refers to the trainable weights used to estimate σ^2. J denotes a joint location coordinate and $\hat{J}$ represents the estimated joint location. The smaller the standard deviation σ is, the more confident the model is with its own estimation. The ground truth coordinates are assumed to follow a Dirac-Delta distribution (i.e. $\sigma \to 0$):

$$P_D(J) = \delta(J - J^{gt}). \tag{5}$$

Our model aims to minimize KL-Divergence between $P_W(J)$ and $P_D(J)$ to learn confidence estimation [16, 38]:

$$\begin{aligned} L_{conf} &= D_{KL}(P_D(J)||P_W(J)) \\ &\propto \frac{e^{-\alpha}}{2}(J^{gt} - \hat{J})^2 + \frac{1}{2}\alpha \end{aligned} \tag{6}$$

where $\alpha \triangleq \log(\sigma^2)$.

In practice, our network predicts the α's using a two-layer neural network $\Psi_{var}(\cdot) = \Psi^2_{var}(\Psi^1_{var}(\cdot))$ (shown as pink blocks in Figure 1):

$$\alpha_{2D}, \alpha_{3D} = \begin{cases} \Psi_{var}(\mathcal{R}(\mathcal{F}(x))), & l = 0 \\ \Psi_{var}(\mathcal{R}(\mathcal{F}(x), \mathcal{M}^l)), & l > 0 \end{cases} \tag{7}$$

Following Eq. (6), the loss for variance estimation of 3D predictions is defined as:

$$L_{var_{3D}} = \frac{e^{-\alpha_{3D}}}{2} L_{3D} + \frac{1}{2}\alpha_{3D}. \tag{8}$$

Since we regress 2D joint locations with a smooth L_1 loss, we can define the loss for variance of 2D joint estimations as done in [11, 16]:

$$L_{var_{2D}} = e^{-\alpha_{2D}}(L_{2D} - \frac{1}{2}) + \frac{1}{2}\alpha_{2D}. \tag{9}$$

Algorithm 1: Progressive Training Protocol

Inputs: maximum number of loops l_{max}, training data $S = \{s_i\} = \{(x, J^{gt}_{2D}, J^{gt}_{3D})_i\}$ and learning rate $\texttt{Lr}$

for $l_{prog} = 0$ *to* l_{max} **do**
- Initialize DIR-Net $\texttt{DIR}_{w,l_{prog}}$ with $l_{prog} + 1$ batch-norm layers
- **if** $l_{prog} > 0$ **then**
  - $\texttt{DIR}_{w,l_{prog}} \leftarrow \texttt{DIR}_{w,l_{prog}-1}$
  - Detach FE from training
  - Reduce $\texttt{Lr}$ by $\frac{1}{10}$ except AMG
- **end**
- **while** *NOT stop criterion* **do**
  - **for** $s_i \in S$ **do**
    - **for** $l = 0$ *to* l_{prog} **do**
      - $\hat{J}^l_{2D}, \hat{J}^l_{3D}, \alpha^l_{2D}, \alpha^l_{3D} = \texttt{DIR}_{w,l}(x)$
    - **end**
    - Calculate L_{total} using Eq. 12
    - Update w based on L_{total} by $\texttt{Lr}$
  - **end**
- **end**

end

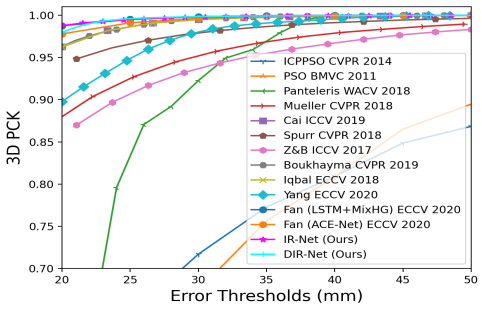
The total uncertainty loss (L_{var}) is defined as the sum of $L_{var_{2D}}$ and $L_{var_{3D}}$. Using L_{var}, we allow the network to learn and estimate variances as a vector, while conventional 3D hand pose estimators use decoders with deconvolutional layers to directly estimate Gaussian heatmaps for estimating the confidence scores [2, 4, 5, 19, 30]. Notably, the exponential terms in the objective function work as adaptive weights between L_{2D} and L_{3D} during training [24, 54].

The estimated variances can be directly utilized for decisions of exiting. We set a threshold value τ_{var} for the average variance for current joint estimations. If the average variance is larger than the threshold τ_{var}, that means the keypoint estimations can be further refined, therefore another loop of RF + AMG is performed. If average variance is lower than τ_{var}, we exit the loop. Figure 4 shows examples of heatmaps generated by the estimated σ^2.

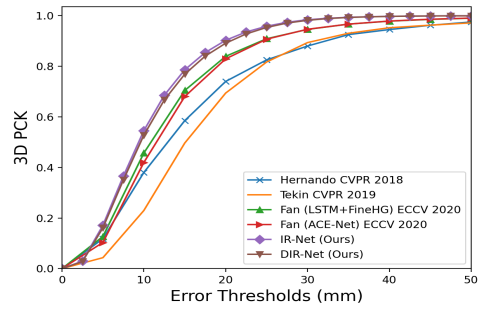
4.2. Decision Gating Function

As an alternative to the heuristic uncertainty threshold value that decides whether to exit or continue, we propose a decision gating function for the network to learn its optimal decisions. This gating function, which is a two-layer neural network, is trained using a reward that optimizes the accuracy-efficiency trade-off. For an input x and attention maps generated at l-th loop $\mathcal{M}^l$, the gating function outputs a *stochastic* categorical decision of exiting.

The gating function $\mathcal{G}$ takes the feature vector $f = \Psi^1_{var}(\cdot)$ from the uncertainty estimation module as inputs. As shown in Figure 1 and Eq. 7, f is a resultant vector created by the input x and attention map generated at the loop $\mathcal{M}^l$ while also being the most determinant factor for variance estimations α. The feature vectors thus consider such information for exit decisions. To this end, we train the gating function $\mathcal{G}(A_l|f_l)$ with on-policy vanilla policy gradi-



(a) PCK for STB



(b) PCK for FPHA

Figure 5: 3D PCK Curve for STB and FPHA

ent updates for two categorical actions $A_l \in \{\text{EXIT, NOT EXIT}\}$. The gradient update is given by:

$$\nabla_w H(w) = \mathbb{E}_{\mathcal{G}}\left[r_l \nabla_w \log \mathcal{G}_w(A_l|\alpha_l)\right], \tag{10}$$

where w represents learnable parameters of the gating neural network, r_l represents the immediate reward signal for the current loop, and $H(\cdot)$ is the total expected reward.

We design the reward signal r_l as a combination of the loss and computational cost (GFLOPs) taken by current iterations:

$$r_l = -\lambda(L^l_{2D} + L^l_{3D}) - l \cdot \text{GFLOPs} \tag{11}$$

where λ is a scale constant. With such rewards, the policy ultimately tries to minimize the error in the pose estimations while also minimizing the computational cost required.

The gating network is trained separately, after the rest of the network including the uncertainty estimation component has been trained. While training the gating module, the remaining parts of DIR-Net are detached from training.

During inference, the gating function outputs a temperature-based softmax distribution [21], from which the actions are sampled. Using the softmax temperature parameter, τ_{gate}, we can control the 'harshness' of the exiting in our network. If τ_{gate} is higher, the model exits in the earlier loop iterations and therefore lesser number of FLOPs are used. As shown in Figure 7, by varying τ_{gate}, we can obtain different accuracy vs computation trade-offs. the parameter τ_{gate} allows expost-facto control over our model

Table 1: SOTA Efficiency Comparison

STB			
Methods	AUC(20-50)	GFLOPs	#Params
Z&B[55]	0.948	78.2	-
Liu et al.[28]	0.964	16	-
HAMR[53]	0.982	8	-
Cai et al.[6]	0.995	6.2	4.53M
Fan et al.[9]	0.996	1.6	>4.76M
Ours	**0.997**	1.31	1.68M
FPHA			
Methods	AUC(0-50)	GFLOPs	#Params
Tekin et al.[45]	0.653	13.62	14.31M
Fan et al. [9]	0.731	1.37	>5.76M
Ours	**0.768**	0.28	460K

after training, which can be used to adjust the generated action probabilities to change the number of loops after training the model.

5. Progressive Training

In the training of our network, we try to minimize the sum of losses from all loops with a predefined maximum number of loops l_{max} [39]:

$$L_{total} = \sum_{l=0}^{l_{max}} \gamma_{2D} L^l_{2D} + \gamma_{3D} L^l_{3D} + \gamma_{var} L^l_{var}. \tag{12}$$

This can be done in either an end-to-end or progressive manner. The end-to-end protocol trains once with a predefined maximum number of loops. The progressive protocol trains the network multiple times while incrementing the total number of loops progressively.

The progressive training protocol is summarized in Algorithm 1. We train the network $l_{max} + 1$ times, each time with the maximum number of loops $l_{prog} \in \{0, 1, 2, ..., l_{max}\}$. The network is initially trained without any loop for the case of $l_{prog} = 0$. This initial training phase requires one BatchNorm layer at the beginning of RF for a single inference path. Then, for every $l_{prog} > 0$, DIR-Net is initialized with $l_{prog} + 1$ number of BatchNorm layers, and the parameters that were learned in the $l_{prog} - 1$ iteration are loaded into the network (except for the extra BatchNorm layer). Since the FE component of the network learns meaningful feature encoding layers when trained with $l_{prog} = 0$, FE is detached from further learning when $l_{prog} > 0$. The learning rate is reduced by a factor of $1/10$ in every iteration for the components of the network, except the AMG which is trained with the original base learning rate. The progressive training protocol is empirically shown to ensure maximization of network's capacity at each loop, yielding a higher frequency of exits at early loops hence lowering average computational cost for inference.

6. Experiments

Datasets. STB dataset has real hand images sequentially captured in 18,000 frames with 6 different lighting conditions and backgrounds. Each frame image is labeled with 3D annotations of 21 joints. Along with the training set of STB, our model is trained with PANOPTIC datasets [40] and evaluated on the testing set of STB, as done in [4]. PANOPTIC dataset is re-engineered from data from multiple views of Panoptic studio [23]. The dataset is made of 14,847 image samples along with 2D joint annotations, and provides general views of hands and skin tones.

FPHA dataset [10] consists of RGB video sequences of 6 subjects performing 45 types of hand activities with daily objects in egocentric views (e.g. pouring a bottle, charging a phone) that follow with heavy (self-)occlusions. Each sample is annotated with 2D and 3D labels both of which are used for training. We follow the official split of the dataset.

Metrics. For evaluation results, we measure the percentage of correct keypoints (PCK) for estimated 2D/3D joint locations and the area under the curve (AUC) of various error thresholds. In addition, we provide average Euclidean distance error for all 2D/3D joints for absolute comparisons.

Implementation Details. Following our main comparison [9], we also present two models that are structured differently for the datasets. Since our work not only improves computational efficiency but also decreases the overall model size, we believe it is more objective to see our models in their smallest size possible that follow with competitive performance. Within IR9, 32 and 16 base channels (i.e. # output channels in first layer) are used respectively for STB and FPHA dataset. For the fully connected layers, the number of nodes for each layer is 512 for STB and 256 for FPHA. The initial learning rate is 10^{-3} for both progressive and end-to-end trainings. For the progressive protocol, the network is initially trained with neither looping nor use of AMG for 50 epochs. For loops $l > 0$, the network is reset with new learning rates as described in Sec. 5 and trained for 20 epochs for each l_{prog} case. For end-to-end training protocol, the network is trained for the same amount of epochs for equivalent maximum loop training setting (e.g. $90 = 50 + 20 + 20$ epochs for $l_{max} = 2$).

6.1. Comparison Against SOTA Methods

Our methods without/with dynamic gating are respectively labeled as 'IR-Net' and 'DIR-Net' for comparison against relevant contemporary methods in Figure 5. For STB dataset, entries include deep-learning based works of [4, 5, 9, 30, 42, 49, 53, 55] and approaches from [33, 52]. Figure 5a shows that our method without adaptive gating performs the best of the entries with 3D AUC of 0.998. With adaptive gating, our method performs with AUC of 0.997 which outperforms the recent adaptive 3D HPE method [9].

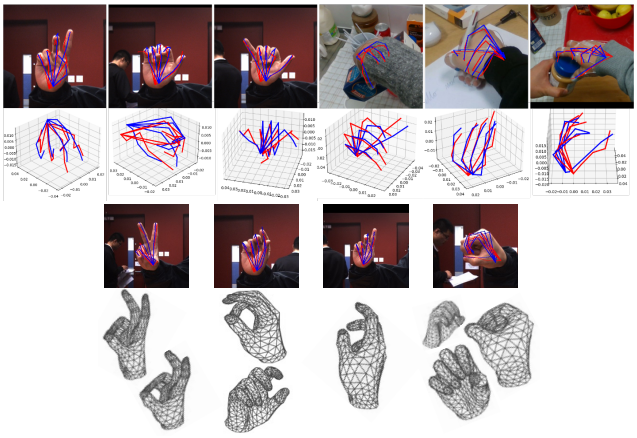


Figure 6: Qualitative results for STB and FPHA datasets

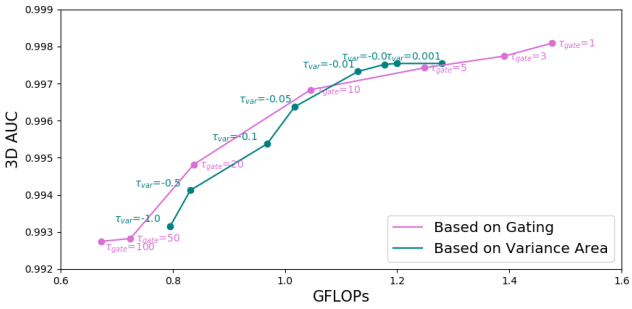
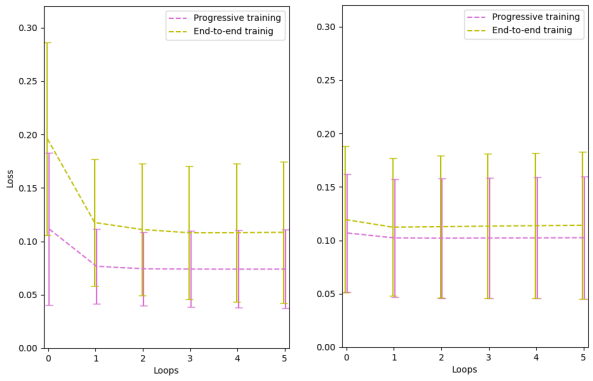


Figure 7: Efficiency and accuracy trade-off for STB with various variance thresholds τ_{var} and temperature parameters τ_{gate}.

For FPHA, our method is compared against contemporary methods including [9, 10, 45], and outperforms them as plotted in Figure 5b. We believe large performance enhancement comes from our effective architecture with residual structure and kinematic fitting of predefined hand model, especially for FPHA dataset, yet our method allows to preserve the accuracy though of large reduction in model size and complexity. Qualitative results for both datasets are depicted in Figure 6 and the Supplementary.

Our model's overall performance is compared to SOTA methods in Table 1. The work of *Fan et al.* [9] is a recent efficient method with attempts of reducing the model complexity for 3D HPE tasks, the size of which is reported partially instead of their whole model. Their coarse and fine pose encoders add up to what is reported in the table, which makes their overall model size even larger. The average of loops required for our method during validation is 4.2 for STB and 2.25 for FPHA.

Computational Efficiency. In Figure 7, we show the efficiency and accuracy trade-off of our iterative refinement model trained for STB dataset. Plots of 3D AUC for for various τ_{gate} represent the temperature parameter of our softmax policy gating outputs. Higher values of τ_{gate} cause softer distribution of softmax. Various heuristic values of



(a) STB (b) FPHA

Figure 8: Range of losses of validation samples after various different numbers of recursive inferences.

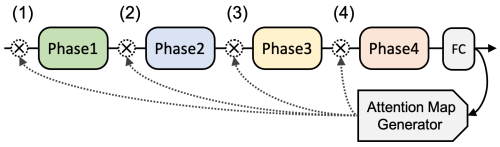


Figure 9: Possible loop points within IR9 structure, which also affects the structure/complexity of FE, RF and AMG.

τ_{var} are explored for the performance trade-offs. Our gating function reaches higher overall performance than that of heuristic thresholds. Meticulous control over τ_{var} is needed to reach competitive performances gained with τ_{gate} values.

6.2. Ablation Studies

For the recursive path of IR9, four possible points can be selected to perform iterative inference as depicted in Figure 9. Since output feature maps vary in sizes among the points, the structure of AMG also changes to generate corresponding spatial maps, which yields different computation cost for each loop point. We provide pose estimation results of our proposed structure with various loop points in Table 2. Here, the results after the final recursive inference of DIR-Net are reported. Although we got the best 3D AUC with the second loop point for FPHA dataset, competitive 2D/3D estimations are achieved with the third loop point with lesser computation, hence providing the best trade-off.

The iterative refinement is evaluated with different structures in Table 3. The baseline is set with our network without any iteration but just forwarding inference. For iterations without AMG, the 7×7 output features after 4th phase of IR9 (see Figure 2b) are spatially enlarged with pixel-shuffle [39] and up-sampling. This method represents direct recursive use of higher-level features as done in [13, 27, 37]. Performances of our network are reported with and without dynamic exits. Results show that recursive refinements effectively fuse higher-level features with lower-level ones for the overall best performance.

Table 2: **Ablation studies on looping points** Prediction results based on 5th recursive inference of DIR-Net trained with $l_{max} = 5$ are reported.

STB

Loop Points	AUC (20-50) 2D	AUC (20-50) 3D	Err (px/mm) 2D	Err (px/mm) 3D	GFLOPs
(1)	0.751	0.993	7.81	8.26	2.22
(2)	0.791	0.994	6.12	7.86	1.63
(3)	**0.806**	**0.998**	**5.93**	**6.88**	1.51
(4)	0.780	0.996	6.69	7.39	0.86

FPHA

Loop Points	AUC (0-50) 2D	AUC (0-50) 3D	Err (px/mm) 2D	Err (px/mm) 3D	GFLOPs
(1)	0.716	0.764	8.68	11.79	0.6
(2)	0.713	**0.775**	8.77	**11.26**	0.5
(3)	**0.717**	0.772	**8.64**	11.54	0.45
(4)	0.716	0.767	8.65	11.68	0.29

Table 3: **Ablation studies on recursive refinement structure**

	STB			FPHA		
	3D AUC	GFLOPs	#Params	3D AUC	GFLOPs	#Params
No Iter.	0.987	0.46	1.44M	0.768	0.14	408K
Iter. w/out AMG	0.991	1.06	1.44M	0.769	0.30	408K
Iter. w/ AMG	0.998	1.51	1.68M	0.772	0.45	460K
Iter. w/ AMG + $\mathcal{G}$	0.997	1.31	1.68M	0.768	0.28	460K

Figure 8 presents the distribution of losses computed at each iteration, implicitly showing the performance difference made by our model trained in progressive and end-to-end manners. End-to-end training of recursive structure carry the conventional gradient vanish problem [3], as also mentioned in a relevant work [13]. Such implication is provided with more comparison results of the progressive and end-to-end training protocols in the Supplementary.

7. Conclusion

In this work, we propose DIR-Net, a lightweight network parts of which are utilized recursively for prediction refinements with adaptive scoping and dynamic gatings. For gating criteria, we introduce an objective function that allows our method to estimate uncertainty of its own 2D/3D pose estimations. This allows dynamic exits for computation efficiency based on pre-defined heuristic thresholds for variance or the decision of the gating function. We also investigate the effectiveness of progressive and end-to-end training protocols for our recursive structure. Training a recursive network in a progressive manner with an increasing number of loop allowance is empirically shown to perform better than training in an end-to-end manner, maximizing capacity of parameters that are recursively exploited. The proposed method reaches the SOTA performance in terms of both accuracy and efficiency.

Acknowledgement This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (2021R1A2C3006659).

References

[1] Kazuyuki Arimatsu and Hideki Mori. Evaluation of machine learning techniques for hand pose estimation on handheld device with proximity sensor. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020. 1

[2] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1067–1076, 2019. 5

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 8

[4] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10843–10852, 2019. 4, 5, 7

[5] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–682, 2018. 1, 3, 5, 7

[6] Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2272–2281, 2019. 1, 6

[7] Hyung Jin Chang, Guillermo Garcia-Hernando, Danhang Tang, and Tae-Kyun Kim. Spatio-temporal hough forest for efficient detection–localisation–recognition of fingerwriting in egocentric camera. *Computer Vision and Image Understanding*, 148:87–96, 2016. 1

[8] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, et al. Depthlab: Real-time 3d interaction with depth maps for mobile augmented reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 829–843, 2020. 1

[9] Zhipeng Fan, Jun Liu, and Yao Wang. Adaptive computationally efficient network for monocular 3d hand pose estimation. In *European Conference on Computer Vision*, pages 127–144. Springer, 2020. 1, 3, 6, 7

[10] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 409–419, 2018. 7

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 4, 5

[12] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. 1, 3

[13] Qiushan Guo, Zhipeng Yu, Yichao Wu, Ding Liang, Haoyu Qin, and Junjie Yan. Dynamic recursive neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2019. 1, 3, 4, 8

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 2

[16] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2888–2897, 2019. 4, 5

[17] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 1, 2, 4

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[19] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 118–134, 2018. 3, 5

[20] Alejandro Jaimes and Nicu Sebe. Multimodal human–computer interaction: A survey. *Computer vision and image understanding*, 108(1-2):116–134, 2007. 1

[21] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 6

[22] Youngkyoon Jang, Seung-Tak Noh, Hyung Jin Chang, Tae-Kyun Kim, and Woontack Woo. 3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. *IEEE Transactions on Visualization and Computer Graphics*, 21(4):501–510, 2015. 1

[23] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015. 7

[24] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 5

[25] Sangho Lee, Simyung Chang, and Nojun Kwak. Urnet: User-resizable residual networks with conditional gating module. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4569–4576, 2020. 2, 3

[26] Sam Leroux, Steven Bohez, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. Resource-constrained classification using a cascade of neural network

layers. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015. 2

[27] Sam Leroux, Pavlo Molchanov, Pieter Simoens, Bart Dhoedt, Thomas Breuel, and Jan Kautz. Iamnn: Iterative and adaptive mobile neural network for efficient image classification. *arXiv preprint arXiv:1804.10123*, 2018. 1, 3, 4, 8

[28] Jun Liu, Henghui Ding, Amir Shahroudy, Ling-Yu Duan, Xudong Jiang, Gang Wang, and Alex C Kot. Feature boosting network for 3d pose estimation. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):494–501, 2019. 1, 6

[29] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2

[30] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018. 3, 5, 7

[31] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018. 2

[32] Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 475–480. IEEE, 2016. 2

[33] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single rgb frame for real time 3d hand pose estimation in the wild. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 436–445. IEEE, 2018. 3, 7

[34] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. User-defined gestures for augmented reality. In *IFIP Conference on Human-Computer Interaction*, pages 282–299. Springer, 2013. 1

[35] Jing Qian, Meredith Young-Ng, Xiangyu Li, Angel Cheung, Fumeng Yang, and Jeff Huang. Portalware: A smartphone-wearable dual-display system for expanding the free-hand interaction region in augmented reality. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, page 1–8, New York, NY, USA, 2020. Association for Computing Machinery. 1

[36] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), Nov. 2017. 4

[37] Oindrila Saha, Aditya Kusupati, Harsha Vardhan Simhadri, Manik Varma, and Prateek Jain. Rnnpool: Efficient non-linear pooling for ram constrained inference. *arXiv preprint arXiv:2002.11921*, 2020. 1, 2, 3, 8

[38] Geonseok Seo, Jaeyoung Yoo, Jaeseok Cho, and Nojun Kwak. Kl-divergence-based region proposal network for object detection. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2001–2005. IEEE, 2020. 5

[39] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 4, 6, 8

[40] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1145–1153, 2017. 7

[41] Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. In-air gestures around unmodified mobile devices. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 319–329, 2014. 1

[42] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–98, 2018. 7

[43] Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *International Conference on Machine Learning*, pages 6166–6175. PMLR, 2019. 2

[44] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016. 2

[45] Bugra Tekin, Federica Bogo, and Marc Pollefeys. H+ o: Unified egocentric recognition of 3d hand-object poses and interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4511–4520, 2019. 1, 6, 7

[46] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018. 1, 2

[47] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018. 1, 2

[48] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018. 2

[49] John Yang, Hyung Jin Chang, Seungeui Lee, and Nojun Kwak. Seqhand: Rgb-sequence-based 3d hand pose and shape estimation. In *European Conference on Computer Vision*, pages 122–139. Springer, 2020. 1, 3, 4, 7

[50] YoungJoon Yoo, Dongyoon Han, and Sangdoo Yun. Extd: Extremely tiny face detector via iterative filter reuse. *arXiv preprint arXiv:1906.06579*, 2019. 1, 2, 3

[51] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 1

[52] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, 2016. 3, 7

[53] Xiong Zhang, Qiang Li, Hong Mo, Wenbo Zhang, and Wen Zheng. End-to-end hand mesh recovery from a monocular rgb image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2354–2364, 2019. 3, 6, 7

[54] An Zhao, Mingyu Ding, Zhiwu Lu, Tao Xiang, Yulei Niu, Jiechao Guan, and Ji-Rong Wen. Domain-adaptive few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1390–1399, 2021. 5

[55] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *Proceedings of the IEEE international conference on computer vision*, pages 4903–4911, 2017. 1, 3, 6, 7