HAND ANALYSIS FROM DEPTH IMAGES

by

MOHAMMAD REZAEI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2022

"*The only guarantee, ever, is that things will go wrong. The only thing we can use to mitigate this is anticipation. Because the only variable we control completely is ourselves.*"

Ryan Holiday

ACKNOWLEDGEMENTS

I want to begin by expressing my deep gratitude to my former advisor, Dr. Ramez Elmasri, who regrettably passed away recently, for supporting me every step of the way. He was more than an advisor to me. I wish he could see this day. I would like to express my deepest appreciation to my former co-advisor and current advisor, Dr. Vassilis Athitsos, whose support and advice have been invaluable over these past years and have surely contributed to my success. He not only greatly helped me with his technical knowledge but also helped profoundly shape my understanding of conducting research. I would also like to thank my academic committee members: Dr. Manfred Huber, Dr. Chris Conly and Dr. Alex Dillhoff for their valuable feedback.

I am grateful to all the teachers who taught me during the years I spent at UTA. I am also thankful to all UTA-CSE faculty/staff members.

Finally, my deep and sincere gratitude to my family for their unconditional love, help and support.

August 11, 2022

ABSTRACT

HAND ANALYSIS FROM DEPTH IMAGES

Mohammad Rezaei, Ph.D.

The University of Texas at Arlington, 2022

Supervising Professor: Vassilis Athitsos

Hand analysis using vision systems is necessary for interaction between people and digital devices and thus is crucial in many applications relating to computer vision and human computer interaction (HCI). The proposed dissertation will explore hand analysis from depth images along two lines: hand part segmentation and 3D hand pose estimation.

First, we investigate hand part segmentation from depth images, which is formulated as a semantic segmentation task. We explore a method aimed at determining for every pixel what hand part it belongs to. This method attempts to perform this task without requiring the ground-truth segmentation labels for training. It uses the 3D hand pose annotations, already provided with hand pose datasets, as a form of weak supervision for training. Both qualitative and quantitative experiments confirm the effectiveness of the proposed method.

Second, we investigate a method that enables accurate 3D hand pose estimation from depth images. This is achieved by a novel formulation of the decomposition of the 3D hand pose estimation into the estimation of 2D joint locations in the depth image space (UV), and the estimation of their corresponding depths aided by

two complementary attention maps. This decomposition prevents depth estimation, which is a more difficult task, from interfering with the UV estimations at both the prediction and feature levels. We empirically show that the proposed formulation of the decomposition of the 3D hand pose estimation and its interaction with two complementary attention maps estimated by the model by two separate branches leads to the state-of-the-art accuracy on three public 3D hand pose estimation benchmark datasets.

Finally, we explore a semi-supervised method for 3D hand pose estimation from depth images. This method is aimed at reducing the reliance of model's training on the ground-truth annotations, which are costly to acquire. This goal is achieved by adopting a student-teacher framework. The teacher network is trained by taking advantage of consistency training and adapting the latest advancements in semi-supervised image classification methods. It generates pseudo-labels for training the student network. As the training progresses, the teacher network improves and generates more accurate pseudo-labels for the training of the student network, resulting in further improvement in the student network. For inference at test time, only the student network is used, and the teacher network is discarded after training. We conduct several experiments to demonstrate the effectiveness of the proposed framework.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

Hands are crucial in allowing humans to interact with the world around them. As such, accurate hand analysis has many applications in areas such as human computer interaction (HCI), augmented reality (AR), virtual reality (VR) and gesture recognition. This dissertation explores two forms of hand analysis: hand part segmentation and 3D hand pose estimation.

As commodity depth cameras become more accurate and affordable, they have become more widely used, resulting in an increased popularity in methods that perform on depth images. Depth-based 3D hand pose estimation and hand segmentation methods have witness significant advancements in the recent years. However, these tasks remain to be very challenging due to the large degree of variation in hand appearance, heavy self-occlusion and noise. This dissertation is aimed at providing novel solutions for the above-mentioned tasks, and is focused on the set of hand analysis methods that process depth images as the input. In this chapter, the primary contributions introduced in this dissertation are briefly reviewed.

1.1 Contributions

This dissertation makes contributions towards two forms of hand analysis from depth images, namely 3D hand pose estimation and hand part segmentation. This section highlights these contributions as they pertain to each topic.

1.1.1 Weakly-supervised Hand Part Segmentation

Existing learning-based methods require a large number of labeled data to produce accurate part segmentation labels. However, acquiring ground truth labels is costly, giving rise to the need for methods that either require fewer labels or can utilize other currently available labels as a form of weak supervision for training. In order to mitigate the burden of labeled-data acquisition, this dissertation proposes a data-driven method for hand part segmentation on depth maps without any need for extra effort to obtain segmentation labels. The proposed method uses the labels already provided by public datasets in terms of major 3D hand joint locations to learn to estimate the hand shape and pose given a depth map. Given the pose and shape of a hand, the corresponding 3D hand mesh is generated using a deformable hand model and then rendered to a color image using a texture based on Linear Blend Skinning (LBS) weights of the hand model. The segmentation labels are then computed from the rendered color image. Since segmentation labels are not provided with current public 3D hand pose datasets, we manually annotate a subset of the NYU dataset to perform quantitative evaluation of our method and show that a mIoU of 42% can be achieved with a model trained without using segmentation-based labels. Both qualitative and quantitative results confirm the effectiveness of our method.

1.1.2 Fully-supervised 3D Hand Pose Estimation

3D hand pose estimation methods have made significant progress recently. However, estimation accuracy is often far from sufficient for specific real-world applications, and thus there is significant room for improvement. In this dissertation, we propose TriHorn-Net, a novel model that uses specific innovations to improve hand pose estimation accuracy on depth images. One innovation is PixDropout, which is, to the best of our knowledge, the first appearance-based data augmentation method

for hand depth images. Another innovation is the use of two complementary attention layers, one for estimating 2D pixel locations of joints, and one for estimating the depth values for those joints. Experimental results demonstrate that the proposed model outperforms the current state-of-the-art methods on three publicly available benchmark datasets.

1.1.3 Semi-supervised 3D Hand Pose Estimation

Despite the significant progress that depth-based 3D hand pose estimation methods have made in recent years, they still require a large amount of labeled training data to achieve high accuracy. However, collecting such data is both costly and time-consuming. To tackle this issue, we propose a semi-supervised method to significantly reduce the dependence on labeled training data. The proposed method consists of two identical networks trained jointly: a teacher network and a student network. The teacher network is trained using both the available labeled and unlabeled samples. It leverages the unlabeled samples via a loss formulation that encourages estimation equivariance under a set of affine transformations. The student network is trained using the unlabeled samples with their pseudo-labels provided by the teacher network. For inference at test time, only the student network is used. Extensive experiments demonstrate that the proposed method outperforms the state-of-the-art semi-supervised methods by large margins.

1.2 Dissertation Structure

This dissertation addresses the two forms of hand analysis in 3 chapters. Each chapter presented in this dissertation is meant to be self-contained, with all relevant background material provided as necessary. In chapter 2, we present a method for performing hand part segmentation. Its objective is to enable the training of such

method without the need for the ground-truth segmentation labels. For training, it uses the currently available datasets that provide 3D joint locations. Chapter 3 is focused on a method for performing fully-supervised 3D hand pose estimation with the state-of-the-art accuracy. Chapter 4 presents a novel framework to perform semi-supervised 3D hand pose estimation, where the goal is to reduce the reliance of the model's training on the ground-truth annotations. This dissertation concludes in chapter 5 with a summery of contributions of each chapter and the potential future research directions.

CHAPTER 2

Weakly-supervised Hand Part Segmentation

2.1 Introduction

Hand part segmentation using vision systems is necessary for interaction between people and digital devices and thus is crucial in many applications relating to computer vision and human computer interaction (HCI). Hand part segmentation is a very challenging task due to the large degree of variation in hand appearance, heavy self-occlusion, large variability in global orientation and self-similarity between hand parts.

As depth cameras become more accurate, more affordable, and more widely used, significant advancements have been made in depth-based hand pose estimation [5, 6, 7, 8, 9, 10, 11, 12] and segmentation [2, 13]. Nonetheless, hand part segmentation has received little attention. In this chapter, we propose the first data-driven method to perform hand part segmentation. Our method differs from existing depth-based hand segmentation methods in that they consider the whole hand as one semantic entity and attempt to segment out the hand from the background [2, 13]. In contrast, we divide the hand into six semantic parts, namely five fingers as well as the palm and attempt to assign pixel-wise labels to the input depth image.

It is widely recognized that deep learning-based methods are data intensive and thus require a large amount of annotated data to learn to carry out their respective tasks. However, acquiring segmentation labels for depth images is costly and labor intensive. To mitigate the burden of labeled data acquisition, our method proposes a framework to use the 3D hand pose labels, already provided with most public

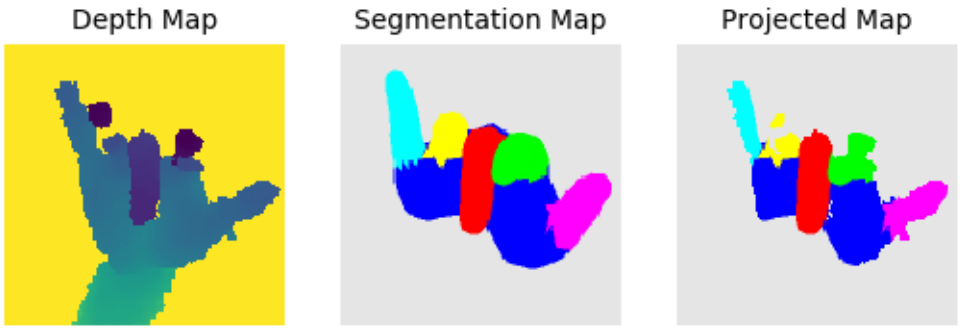

Figure 2.1: The outline of the the proposed method

datasets, as a form of weak supervision. More specifically, a deep model is first trained to perform both 3D hand pose and shape estimation similar to [14]. The hand shape is represented as a triangular mesh parameterized by pose coefficients of a deformable hand model [15]. As a preprocessing step, we use LBS weights to assign each triangular face of the mesh a semantic label that determines which hand part it belongs to. Each hand part is given a pre-defined color. Based on this color map, we create a texture by which the mesh is rendered to a color image. We can simply obtain each pixel's semantic label according to their color. Finally, the color image is refined to ensure that the segmentation labels are aligned with the input depth map (see Fig. 2.1).

In recent years, data-efficient hand pose estimation methods have gained popularity as models have grown larger and more complicated, resulting in the significantly increased need for labeled training data. Authors in [16, 17] use different data modalities to compensate for the lack of available annotated data. In [18], 2D annotations are used as weak supervision to train a 3D pose estimator. Wan in [19] proposed a data-driven self-supervised method for the task of depth-based 3D hand pose estimation to eliminate the need for any real data label. Our method is similar in spirit to

these methods as it is aimed at mitigating the need for explicit labeled data which could be hard to acquire.

Despite some similarities, our method differs from [14] in that their goal is to estimate hand shape and pose, while our method is aimed at performing hand part segmentation. Our method also estimates hand shape and pose as a by-product. We conduct extensive experiments to evaluate the proposed method both qualitatively and quantitatively. Since there is no public depth dataset that provides both hand part segmentation labels and 3D joint locations as ground-truth, we manually label a subset of the NYU dataset to perform quantitative evaluation.

To the best of our knowledge, this is the first data-driven method to perform hand part segmentation on depth images. Both quantitative and qualitative results confirm that our method achieves a good performance despite the fact that it has not been trained with a single segmentation-labeled data.

2.2 Hand Segmentation

Most existing methods cast hand segmentation as a dense prediction problem for every pixel in the image, where the task is to assign every pixel a label to determine whether it belongs to hand or not (binary classification). Hand segmentation from color images can be broadly categorized into two groups: 1). methods that take their visual clue from and are based on skin [20, 21, 22, 23]. One can refer to [24] for more details. 2). methods that are based on motion [25, 26, 27, 28, 29].

However, it is notoriously challenging to accurately detect skin in unconstrained settings due to severe light condition variations and complex effects like subsurface scattering, making it difficult to develop segmentation methods that could work well on images in the wild. Unlike color images, hand segmentation from depth images does not suffer from these problems. This line of research was pioneered by [2].

[13] provided a dataset for hand segmentation on depth images featuring multiple hands. In [30], they proposed a method to perform hand segmentation for hand-object interaction.

In contrast to these methods, we formulate our problem as a semantic segmentation task where the goal is to assign one label from a predefined set of class labels (one label per part) to each pixel [31]. In other words, we are interested in determining for every pixel what hand part it belongs to.

In [32, 33], they perform hand part segmentation as part of their experiment and use its performance as a proxy for the accuracy of 3D pose estimation. [34] reports hand part segmentation performance as a proxy for detailed surface registration. However, our method is fundamentally different from them in that their approach is not data-driven, meaning that these methods perform optimization on each test data individually, while our method is data-driven and accumulates knowledge over the course of training. Another clear advantage of our data-driven approach is that unlike these methods, it does not require to do computationally expensive optimization at inference time and the inference can be done by a single forward pass of the network, which only takes around 100 ms on average on a single Nvidia GTX 1080 Ti GPU.

2.3 Hand Models

In order to represent the hand, many hand models have been proposed in recent years. Some early works modeled the hand using geometric primitives [35]. Subsequent researches used various methods such as sphere meshes [36], sum of Gaussians [37], or loop subdivision of a control mesh [38]. The proposed method uses the hand model proposed in [15], referred to as MANO. The MANO hand model has a high representation power and has made many improvements on the previous hand models including learning pose dependent corrective blend shapes, first proposed in [39], to

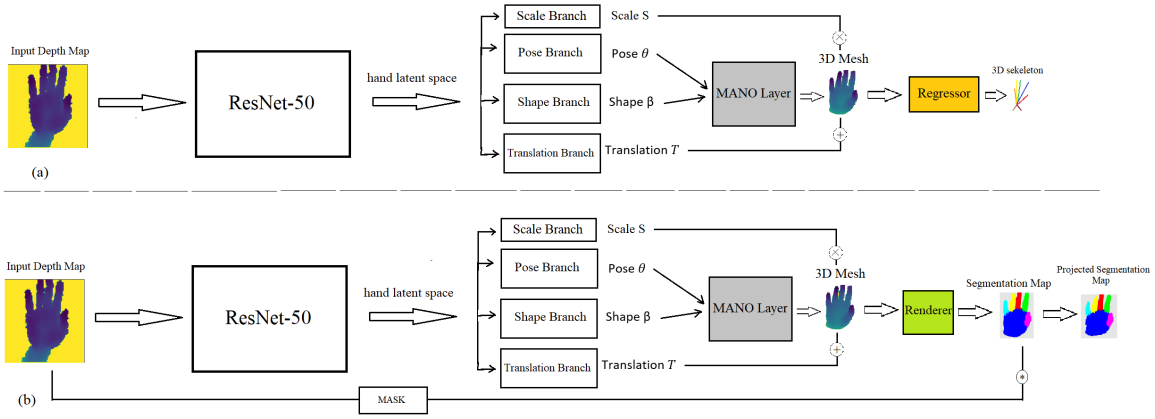

Figure 2.2: The general overview of the method. (a) At training time, the pose and shape estimator network takes a depth image as input and estimates the hand pose, shape, scale and translation, which are used to generate, scale and then translate the corresponding 3D mesh. The 3D mesh serves as input to the Regressor to compute the 3D hand joints. The weak supervision is applied to the estimated 3D joints using the ground-truth provided by the dataset. (b) At inference time, given a depth map, the model estimates the corresponding hand mesh and uses a renderer to obtain a color image. The segmentation labels are then computed based on the color of pixels in the rendered image.

correct some limitations of the standard Linear Blend Skinning that lead to unnatural results. It is used as a fully differentiable layer in our network to allow for end-to-end training of the whole pipeline.

2.4 Methodology

As illustrated in Fig. 2.2, our method includes two general stages. The first stage essentially follows the standard training paradigm of 3D pose estimation methods [14]. The 3D pose and shape estimator network is trained using weak labels in terms of 3D joint locations of the hand. The network takes as input a depth map of size 128×128 and regresses the hand shape $\overrightarrow{\beta}$ and pose $\overrightarrow{\theta}$ parameters, scale S and the translation vector T. The hand parameters are passed on to the differentiable articulated mesh deformation hand model that generates a triangulated 3D mesh. The 3D mesh is

scaled by S and then translated by T, which is in turn fed to the regressor to compute the underlying 3D skeleton. The supervision is applied to the predicted 3D skeleton to minimize its deviation from the ground-truth 3D skeleton. The regressor is a single linear layer trained prior to this stage and kept fixed over the course of training at this stage. The hand model is also kept fixed during training. At inference time, the network takes a depth image as input and estimates the hand model parameters to generate its corresponding 3D mesh. The 3D mesh is then rendered to a color image using Neural Renderer [40]. The color of each mesh triangular face is chosen according to the hand part it belongs to, which is derived from LBS skinning weights provided by the MANO hand model. Thus, the semantic label for each pixel can simply be computed based on its color in the rendered color image. Finally, the rendered color image is aligned with the input depth map to ensure that no background pixel is assigned a label as a hand part.

2.4.1 Hand Model

Our model attempts to fit the MANO hand model [15] to the input depth image. The MANO hand model parametrizes a hand using pose parameters $\overrightarrow{\theta}$, which represent the relative rotation between pre-defined joints and their parent joints in the kinematic tree, and hand shape parameters $\overrightarrow{\beta}$, which denote the linear shape coefficients that represent offsets from the template mesh $\bar{T}$. Given hand shape $\overrightarrow{\beta}$ and pose $\overrightarrow{\theta}$ vectors, the template mesh $\bar{T}$ is first sculpted as follows [15]:

$$T_P(\overrightarrow{\beta}, \overrightarrow{\theta}) = \bar{T} + \sum_{n=1}^{|\overrightarrow{\beta}|} \beta_n S_n + \sum_{n=1}^{9K} (R_n(\overrightarrow{\theta}) - R_n(\overrightarrow{\theta^*}))P_n \tag{2.1}$$

where S_n is the n-th principal component of shape displacement, $|\overrightarrow{\beta}|$ is the number of linear shape coefficients, R_n denotes the part relative rotation matrix for the n-th joint in the kinematic tree, $\overrightarrow{\theta^*}$ represents the rest pose, and P_n is the n-th

element in the matrix of pose blend shapes. The sculpted mesh is then deformed using Linear Blend Skinning [41] to generate the 3D mesh as follows [15]:

$$M(\overrightarrow{\beta}, \overrightarrow{\theta}) = W(T_P(\overrightarrow{\beta}, \overrightarrow{\theta}), J(\overrightarrow{\beta}), \overrightarrow{\theta}, \tilde{W}) \tag{2.2}$$

where W is a linear blend skinning [41] function applied to sculpted mesh T rigged with a kinematic tree of 16 joints. J is a joint regressor that takes the template mesh sculpted only by shape blend shapes (before applying pose blend shapes) and regresses the 3D joint locations, and $\tilde{W}$ is the matrix of the LBS weights. The MANO hand model parameters $\bar{T}$,S,P,J and $\tilde{W}$ are learned using registered hand scans by the training procedure detailed in [15]. These parameters are kept fixed during our training process.

In order to reduce the space of pose parameters and thus the possibility of generating unnatural meshes, instead of directly using pose parameters that represent angles between joints and their parents, we use coefficients of Principal Component Analysis (PCA), as in [15], which are computed on angle-axis representation of the respective joints in the data collected to build the model [15]. We use 26 PCA coefficients to represent the hand pose concatenated by a vector of size 3 representing the hand global orientation in axis-angle representation to form the pose vector $\overrightarrow{\theta} \in \mathbb{R}^{29}$. We use 10 coefficients for the shape $\overrightarrow{\beta} \in \mathbb{R}^{10}$.

Given the shape $\overrightarrow{\beta}$ and pose $\overrightarrow{\theta}$ parameters, the MANO layer generates a hand mesh through the function $M(\overrightarrow{\beta}, \overrightarrow{\theta})$ of $N = 778$ vertices and 1538 faces.

2.4.2 Regressor

In order to extract joints that are compatible with the 14 standard joints in the NYU dataset, we pretrain a single-layer feed forward network without activation layer $\bar{R}$ that takes as input a 3D hand mesh and outputs 14 3D joint locations of

Figure 2.3: The hand divided into six semantic parts, which are five fingers as well as the palm

the hand. We train the regressor $\bar{R}$ using manually annotated randomly generated meshes by sampling from the pose $\overrightarrow{\theta} \in [-1.3, +1.3]^{29}$ and shape $\overrightarrow{\beta} \in [-0.01, +0.01]^{10}$ of the MANO hand model. The regression is essentially a matrix multiplication and is therefore fully differentiable and can be integrated into our end-to-end trainable pipeline. After pre-training, the parameters of the regressor are kept fixed for the subsequent training of our pose and shape estimator network.

2.4.3 Pose and Shape Estimator Network

The pose estimator takes as input the depth image and estimates hand pose $\overrightarrow{\theta}$ and shape $\overrightarrow{\beta}$ parameters as well as translation $T = (T_x, T_y, T_z)$ and scale S. The backbone is a ResNet-50 network [42]. Its last fully connected layer is replaced with a fully connected layer of size 256 to encode the hand features into a latent space, followed by four separate branches to estimate the hand pose $\overrightarrow{\theta} \in \mathbb{R}^{29}$, hand shape $\overrightarrow{\beta} \in \mathbb{R}^{10}$, translation $T \in \mathbb{R}^3$ and scale $S \in \mathbb{R}$ respectively (see Fig. 2.2).

2.4.4 Renderer

At inference time, we use the estimated mesh generated by the model to render it to a color image I' using the renderer proposed in [40]. We use a simple texture for rendering which is computed as follows. We determine for each vertex in the mesh what hand part it belongs to based on LBS skinning weights $\tilde{W}$ provided by the MANO hand model. Using this information, we determine for each triangular face of the mesh what hand part it belongs to by doing majority voting among its three vertices. Finally we assign each face a pre-defined color based on what hand part it belongs to (see Fig. 2.3). This process is done offline and needs to be done only once. The segmentation label for each pixel is then easily computed based on the color of each pixel in the rendered color image. Orthographic projection is used for rendering the mesh.

2.4.5 Alignment

When the mesh is rendered to a color image, it may not be fit to the input depth map due to inaccuracies in the estimation of the pose and shape estimator network. This may result in some false positives. For example, some background pixels in the input depth image may be assigned a label as a hand part. In order to prevent this issue, we first compute the foreground mask for both input depth map I and the rendered color image I' as follows:

$$I'_{Mask}(P) = \begin{cases} 1, & \text{if } P_c \neq BC \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

$$I_{Mask}(P) = \begin{cases} 1, & \text{if } P_c \neq BD \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

$$MASK_{ref}(P) = I_{Mask}(P) \wedge I'_{Mask}(P) \tag{2.5}$$

where P denotes pixel, P_c is the value of the pixel P, BC denotes the background color in I' and BD denotes the background depth in the input depth map I. $\wedge$ denotes the logical AND operation. Finally, the rendered color image I' is aligned by multiplying I' by $MASK_{ref}$ to exclude the pixels in I' that correspond to the background pixels in the input depth map:

$$I'_{aligned}(P) = I'(P) \wedge MASK_{ref}(P) \tag{2.6}$$

2.4.6 Training

Since there is no segmentation label at training time, our method uses the 3D joint locations of the hand already provided with most depth-based public datasets to learn the task of hand pose and shape estimation. The pose and shape estimator network is trained by minimizing the following loss:

$$L = \alpha_{joint} L_{joint} + \alpha_{pose} L_{pose} + \alpha_{shape} L_{shape} \tag{2.7}$$

L_{joint} is aimed at minimizing the difference between the estimated joints and the ground-truth joints and is computed as follows:

$$L_{joint} = |J - J'|^2 \tag{2.8}$$

where $J, J' \in \mathbb{R}^{14\times3}$ are the ground-truth and estimated 3D locations of the standard 14 joints respectively. The estimated joint J' is computed as follows:

$$J' = \bar{R}(SM(\overrightarrow{\beta}, \overrightarrow{\theta}) + T) \tag{2.9}$$

where $\overrightarrow{\beta}, \overrightarrow{\theta}$, T and S are shape, pose, translation vector and scale respectively, which are estimated by the pose and shape estimator. L_{pose} and L_{shape} are defined as follows:

$$L_{pose} = \| \overrightarrow{\theta} \| \tag{2.10}$$

$$L_{shape} = \| \overrightarrow{\beta} \| \tag{2.11}$$

where $\|.\|$ denotes Frobenius Norm. L_{pose} and L_{shape} are used to regularize the space of pose and shape parameters to push them to be close to the mean shape and pose, and in doing so encourage generating more physically plausible meshes. To apply sutiable balance between the loss terms, we set $\alpha_{joint} = 10$, $\alpha_{pose} = 1$ and $\alpha_{shape} = 1000$.

2.5 Experiments and Discussion

In this section, we present both quantitative and qualitative results of the proposed method. To evaluate the proposed method, we need a dataset that provides 3D joint locations (for training) and segmentation labels (for testing). However, to the best of our knowledge, there is no such dataset. The only public depth dataset that provides segmentation labels for hand parts is the FingerPaint dataset [32]. However, it does not provide 3D joint locations required for training our method.

Because of the above-mentioned reasons, we chose to train our model on the NYU pose dataset [2], which is one of the most commonly used public benchmarks for hand pose estimation methods. This dataset, captured by 3 calibrated and synchronized PrimeSense depth cameras, consists of 72757 depth images for training and 8252 depth images for testing. NYU is a challenging dataset featuring hands that cover a wide range of hand poses. It provides the labels for depth images in terms of 3D joint locations of the hand, which are used by the proposed method to train the pose and shape estimator. Our network is implemented by PyTorch [43] and the Nvidia GTX 1080 Ti GPU is used for training. The model is trained end-to-end for 40 epochs using Adam optimizer with a learning rate of 10^{-4} and a learning decay of 10^{-1} every 20 epochs.

Table 2.1: Performance in terms of 2D Keypoint localization on the NYU dataset (Finger joints only). Mask R-CNN keypoint refers to the case where joint positions are localized by finding the positions of joint confidence maps with maximum probabilities. Mask R-CNN keypoint and mask restricts keypoints lying on estimated masks

Methods	Mean Keypoint error (Pixels)
Ours	**10.24**
Duan-KNN[44]	10.32
Mask RCNN(kpt and mask)[44]	15.70
Mask RCNN(kpt only)[44]	20.97

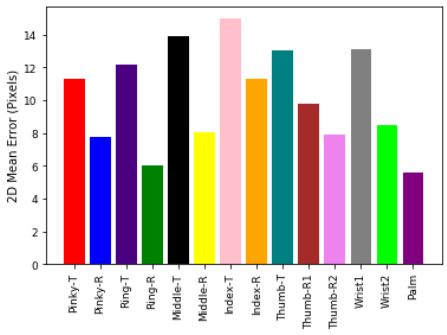

Figure 2.4: Per-joint mean 2D error. T and R denote tip and root respectively (e.g. Index-T denotes the tip of the index finger)

The NYU dataset provides about 7k annotations for hand segmentation. However, they are not suitable for our evaluation since they provide binary labels (hand or none-hand), whereas our method needs part-based segmentation labels. Thus, we manually label a subset of size 500 from the NYU test set for quantitative evaluation.

2.5.1 Quantitative Evaluation

We begin the evaluation by reporting the performance of the proposed method in terms of 2D keypoint localization and 3D hand pose estimation. In order to generate accurate segmentation maps, it is crucial for the model to detect hand parts accurately. Thus, the ability of the proposed method to accurately localize 2D hand joints is strongly correlated with the performance of the method in terms of hand

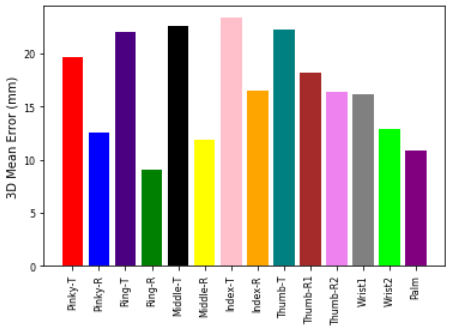

Figure 2.5: Per-joint mean 3D error. T and R denote tip and root respectively (e.g. Index-T denotes the tip of the index finger)

Table 2.2: Performance comparison with some of the state-of-the-art methods in terms of 3D hand pose estimation on NYU dataset [2]

Methods	Mean 3D error (mm)
Ours	17.66
DeepPrior [45]	20.75
DeepPrior-Refine [45]	19.72
DeepModel [46]	17.03
Feedback [47]	15.97
DeepHPS [48]	14.41
3DCNN [49]	**14.11**

part segmentation. As can be seen in Table 2.1, despite the fact that our method's original goal is not to perform 2D keypoint localization, our method outperforms the state-of-the-art methods that were originally used to do 2D joint localization. Since [44] reports the results for only finger joints, in order to have a fair comparison, we select only finger joints out of the standard 14 joints estimated by our method. The numbers reported in Table 2.1 are computed by taking average across the selected joints. The localization accuracy for individual joints can be seen in Fig. 2.4.

The performance of our method in terms of 3D hand pose estimation is reported in Table 2.2. As can be seen, our method compares strongly with the state-of-the-art methods in 3D pose estimation despite the fact that it has not been specialized for

this task. A commonly used metric for reporting 3D results is mean 3D error, which is the average distance between the predicted joint location and its corresponding ground-truth in 3D space. Table 2.2 reports the average across all 14 joints for each method.

Next, we perform evaluation of the segmentation performance of the proposed method. Since we cast our problem as semantic segmentation with 6 classes (five fingers and the palm), we use two commonly used metrics for evaluating semantic segmentation methods. It is worth noting that unlike many RGB-based semantic segmentation methods that consider the background pixels as a separate semantic entity and assign a separate label to them, we do not consider the background pixel label assignment as it is trivial in depth images to segment out the background pixels using simple pre-processing steps such as thresholding. The first metric used for evaluation is Pixel Accuracy, which represents the proportion of pixels in the image that are labeled correctly. The second metric is Intersection over Union (IoU), which is calculated separately for each class, defined follows:

$$IoU = \frac{|TP|}{|TP| + |FP| + |FN|} \tag{2.12}$$

Where TP, FP and FN denote true positive, false positive and false negative respectively. As in [50], to account for class imbalance, we report class-wise average among classes, that is, mean IoU denoted by (mIoU). The results in terms of mIoU can be seen in Table 2.3. It should be mentioned that since this is the first data-driven method proposed to perform depth-based hand part segmentation, there is no other work to compare against.

The hand palm is arguably easier to segment than other parts owing to the fact that fingers are more likely to be occluded, and if they do, the model is likely to mistake one finger for another since they look similar in many cases, which makes

Table 2.3: Hand part segmentation performance

Hand Part	mIoU
Pinky Finger	0.38
Ring Finger	0.41
Middle Finger	0.41
Index Finger	0.37
Thumb	0.39
Palm	0.53
Average	0.42

the task of segmenting fingers more challenging. Fingers also have higher levels of articulation and motion which leads to 3D joint labels of fingers being less accurate in comparison to the palm. As can be seen in Fig. 2.5 and Fig. 2.4, the accuracy of the method to localize the palm keypoints is higher than that for all the finger joints (except for Ring-R in the 3D case). As can be seen in Fig. 2.5 and Fig. 2.4, fingertips are the most difficult keypoints for the model to predict because they tend to get occluded more frequently than other keypoints. Yet, our method achieves a mIoU of 0.39 for fingers. It should be kept in mind that the proposed method achieves a good performance despite the fact that it has not been trained using segmentation labels. We also report the IoU averages across all classes and the Pixel Accuracy to be 0.42 and 92% respectively.

2.5.2 Qualitative Evaluation

In order to verify the quality of the generated segmentation maps and the robustness of the proposed method in various cases, we draw some relatively hard samples from the NYU testing set and show the result of the proposed method on them as illustrated in Fig. 2.6. Experiments demonstrate that our method is capable of generating high-quality hand meshes and as a result accurate part segmentation,

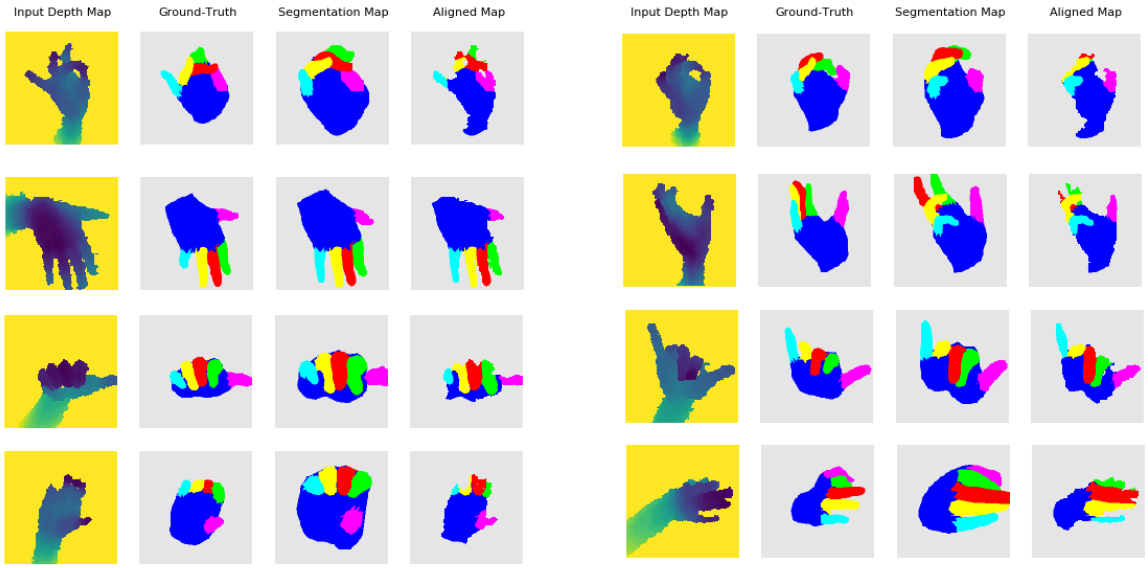

Figure 2.6: Qualitative results of the method given depth images of hands in various poses

which has many potential applications including in animation. Furthermore, as can be seen in Fig. 2.6, the proposed method is robust in estimating hand shape and pose and as a result the segmentation map accurately even in hard scenarios such as self-occlusion and exaggerated articulations.

CHAPTER 3

Fully-supervised 3D Hand Pose Estimation

3.1 Introduction

Accurate 3D hand pose estimation has many applications in areas such as human computer interaction (HCI), augmented reality (AR), virtual reality (VR) and gesture recognition. Despite significant progress made by 3D hand pose estimation methods [5, 6, 7, 8, 9, 10, 11, 12], 3D hand pose estimation remains a very challenging task due to the large degree of variation in hand appearance, heavy self-occlusion, noise, high dimensionality and self-similarity between hand parts [8, 51, 10, 52]. As a result, pose estimation accuracy is often far from sufficient for specific real-world applications, and thus there is significant room for improvement. In this chapter, we propose TriHorn-Net, a novel model that uses specific innovations to improve hand pose estimation accuracy on depth images. The first innovation is decomposition of the 3D hand pose estimation into the estimation of 2D joint locations in the depth image space (UV), and the estimation of their corresponding depths aided by two complementary attention maps. This decomposition prevents depth estimation, which is a more difficult task, from interfering with the UV estimations at both the prediction and feature levels. The second innovation is PixDropout, which is, to the best of our knowledge, the first appearance-based data augmentation method for hand depth images. Experimental results demonstrate that the proposed model outperforms the state-of-the-art methods on three public benchmark datasets.

With the advancement of deep neural networks (DNNs), DNN-based hand pose estimation techniques rapidly displaced the previous methods such as [53, 35] and

have achieved impressive results. These methods can be broadly categorized into two groups: 1) regression-based methods and 2) detection-based methods. Regression-based methods [51, 45, 4] encode the hand depth image into a single global feature which is in turn used to directly estimate the hand joints. Detection-based methods adopt a dense-prediction approach, where they utilize hierarchical features to compute pixel-wise predictions for each joint. Detection-based methods generally tend to outperform regression-based methods [5] because regression-based methods use a single global feature for estimation, which cannot fully retain fine-grained spatial information required for accurate mapping into 3D hand poses.

Despite the superior performance of detection-based methods, they still suffer from several drawbacks. Moon et al. [52] achieve a high accuracy by using 3D CNNs, but their method comes at a heavy computational and memory cost. A new class of detection-based methods has recently emerged that adopts an approach based on dense pixel or point offset prediction, whereby they densely estimates all the pixels'(or points') offsets to joints and compute joint positions by a weighted average over all the corresponding offset values. Despite their high performance, [54, 55] use non-learnable information aggregation operations such as argmax operation or mean-shift estimation to compute joint coordinates from the heatmap or offset vector fields. However, the information aggregation operation is treated as a post-processing step and is not incorporated into the training phase, causing a gap between training and inference. [11, 51] require complex post-processing operations, such as taking neighboring points, causing inevitable quantization errors and rendering the pipeline not end-to-end differentiable. JGR-P2O [56] partly solves these issues by predicting pixel-wise offsets and a weight map to compute the joint positions using the weighted average, but it still suffers from the common issue among this class of methods that

the estimations are unstable when depth values near the target joint are heavily missing, leading to a performance degradation.

To tackle the aforementioned issues, we propose a novel model, that we call TriHorn-Net, for 3D hand pose estimation. TriHorn-Net consists of an encoder network that encodes the input hand depth image into a high-resolution feature volume, and three separate branches that take the hand feature volume as the input and together estimate the 3D hand pose. The first two branches compute two per-joint attention maps that are fused subsequently. The two attention maps are complementary in the sense that one guides the network's attention towards the pixels where the joints occur, and the other guides the network's attention towards non-joint pixels that can potentially give the network useful clues for estimating the corresponding joint depths. The attention maps computed by the UV branch are explicitly encouraged to focus on joint pixels by applying 2D supervision to the heatmaps resulted from passing them through a spatial softmax layer [57]. This approach can be viewed as the typical detection-based approach based on dense pixel-wise joint predictions. The attention maps computed by the attention enhancement branch are learned under no constraints, allowing them to freely focus on hand pixels most relevant to the estimation of joint depths. The depth branch develops pixel-wise feature vectors that contain depth information of the joints. The proposed model uses the fused per-joint attention maps as guidance to pool features from relevant pixels for each joint. After the relevant features are pooled for each joint, a weight-sharing linear layer is used to estimate the corresponding depth value.

We also propose PixDropout, a simple yet effective appearance-based data augmentation function for depth-based hand pose estimation methods. This function performs augmentation on a given sample by uniformly sampling a fraction of the pixels on the hand surface and turning them into a background pixel (replaces their

value with a constant background value). We show empirically that PixDropout leads to a performance improvement not only in the proposed method but also in a regression-based method.

The proposed model is end-to-end differentiable and does not include any post-processing step or data pre-processing such as converting the depth map into point clouds [7, 51, 58] or voxelized volume[52]. We conduct the evaluation of the proposed model on three publicly available datasets, namely ICVL [1], MSRA [3] and NYU [2], which are challenging benchmarks commonly used for evaluation of 3D hand pose estimation methods. The results demonstrate that the proposed model outperforms the state-of-the-art methods on all these benchmarks.

In summary, the contributions in this chapter are as follows:

- We propose a novel neural network architecture, TriHorn-Net, which enables accurate 3D hand pose estimation.
- We propose a novel formulation for effective decomposition of the hand pose estimation into the estimation of the 2D joint locations and their depths.
- We propose PixDropout, which is, to the best of our knowledge, the first appearance-based data augmentation function for depth-based hand pose estimation methods.
- We conduct extensive experiments to demonstrate that the proposed method outperforms the state-of-the-art methods.

3.2 3D Hand Pose Estimation

Hand pose estimation has been a long-standing problem in Computer Vision. Before the widespread use of deep learning techniques, many approaches relied on hand-crafted features, optimization methods, and distance metrics. Athitsos et al. [59] used edge maps and Chamfer matching to perform 3D hand pose estimation.

Other approaches used optimization methods such as Particle Swarm Optimization (PSO) [32, 35]. After the rise of deep learning, DNN-based methods quickly displaced the traditional methods. The two most common input data modalities for DNN-based methods are: 1) RGB images and 2) depth images. While DNN-based 3D hand pose estimation on RGB images is a relatively new field of research, it has attracted a lot of attention recently [60, 61, 62, 63, 64, 65, 66, 67, 68]. However, as our method is depth-based, we focus our attention here on other depth-based methods.

Depth-based hand pose estimation methods have significantly advanced in the last decade. These methods can be classified into three categories: generative methods [38, 15, 69, 70], discriminative methods [53, 71], and hybrid methods [32, 72, 33]. Oberweger et al. [45] used a CNN to estimate the hand pose represented by PCA coefficients. Instead of performing in the 2.5D space, several methods [49, 52] converted 2.5D depth images into 3D voxels and adopted 3D CNNs to estimate the 3D hand pose. Fang et al. [56] propose an approach based on graph CNNs to compute pixel offsets to the joints and use weighted average to compute the hand joint locations. Another line of research has recently emerged, that utilizes the latest advancements of point cloud processing, by converting depth images into point clouds and using a point cloud processing network to perform hand pose estimation [51, 10, 54, 7, 58, 73].

TriHorn-Net is inspired by the methods based on dense pixel-wise prediction, but it differs from them in some important aspects. It offers a novel formulation for hand pose estimation, which is based on the decomposition of hand pose estimation into estimating 2D joint locations and their depth values. While it takes advantage of the typical pixel-wise prediction approach for estimating the 2D joint locations, it breaks from the standard approach in the sense that it estimates pixel-wise feature vectors (as opposed to predictions) and uses a weight-sharing layer (as opposed

to dedicated layers) for estimating the joint's depth values. Extensive experiments demonstrate the effectiveness of the proposed formulation.

The proposed model is similar to A2J [9] in that it uses different branches for estimating joints' image coordinates and their depth values. However, it adopts a fundamentally different approach for estimating the joint positions. A2J [9] relies on a fixed number of regularly spaced points placed in depth image space, which are called anchors, in order to predict joint UVD offsets, whereas the proposed method performs a pixel-wise likelihood estimation for computing joints' image coordinates UV, and uses the UV estimations to guide the estimation of the joint depth values.

3.3 Data Augmentation

Data augmentation methods aim at increasing the amount and diversity of the training data by randomly creating novel and realistic-looking data samples. In recent years, significant progress has been made on data augmentation methods for vision [74, 75], NLP [76] and speech [77, 78]. In the image domain, novel samples are created by applying a set of transformations to an available sample. These transformations can be broadly categorized into two groups: 1) geometric transformations and 2) appearance transformations. There has been a wide range of appearance transformations proposed recently for performing data augmentation on RGB images, such as color jitter, histogram equalization and contrast adjustment. However, most of them are not applicable to depth images due to the different nature of the depth image, limiting the set of data augmentation functions used by depth-based hand pose estimation methods mostly to geometric transformations such as random rotation, translation and scaling.

In this chapter, we propose PixDropout, a simple yet effective appearance transformation applicable as a data augmentation function to the depth images. It is

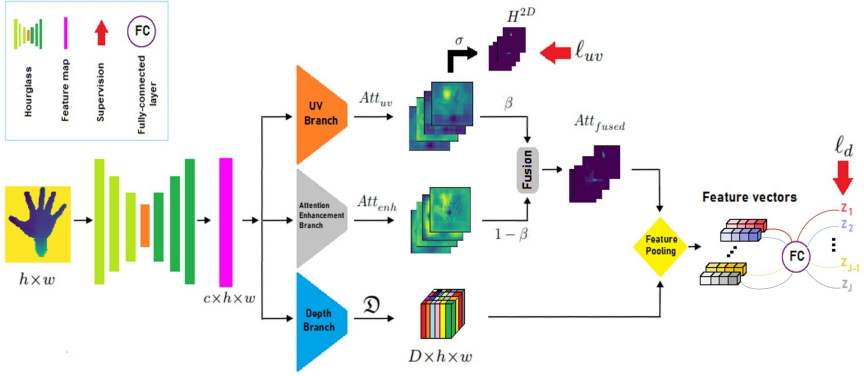

Figure 3.1: An overview of the TriHorn-Net architecture. It consists of an encoder that encodes the hand depth image into a high resolution feature volume, which serves as the input to three separate branches. The UV branch and the attention enhancement branch compute two per-joint attention maps Att_{uv} and Att_{enh} respectively. Att_{uv} is focused on joint pixels, whereas Att_{enh} attention map has the flexibility to shift the network's attention to hand pixels that are most relevant for joint depth estimation. Att_{uv} and Att_{enh} are fused via a linear interpolation controlled by per-joint learned parameters β_j. For estimating the joints' depths, the network uses the fused attention maps Att_{fused} to pool features from the depth feature map $\mathfrak{D}$ computed by the depth branch. The pooled feature vector for each joint is input to a weight-sharing linear layer to estimate its depth value.

strongly inspired by Dropout [79]. While Dropout [79] randomly selects and drops neurons in the layers of a neural network, the proposed PixDropout randomly drops some fraction of pixels on the hand surface in the input depth image. PixDropout is most similar to the RGB image augmentation methods proposed in [80, 81], where they randomly sample and then mask out rectangular regions of the input image to simulate occlusion for an image recognition task. However, in contrast to [80, 81], PixDropout applies no spatial priors (e.g., it samples individual pixels rather than rectangular regions). Empirical results show that despite its simplicity, PixDropout is an effective data augmentation function for depth-based hand pose estimation methods.

3.4 Methodology

The task of 3D hand pose estimation is defined as follows: given an input depth image $D_I \in \mathbb{R}^{H \times W}$, the task is to estimate the 3D location of a set of pre-defined hand joints $P \in \mathbb{R}^{J \times 3}$ in the camera coordinate system. H, W denote the height and width of the input depth image respectively. J denotes the total number of joints to be estimated. In this section, the proposed model is laid out in details.

As illustrated in Fig. 3.1, TriHorn-Net consists of two stages. In the first stage, the input depth image is run through the encoder network f. The encoder extracts and combines low-level and high-level features of the hand and outputs a high resolution feature volume, which is passed on to three separate branches. The UV branch, computes a per-joint attention map, where each map is focused on pixels where the corresponding joint occurs. This behavior is explicitly enforced by the application of 2D supervision to the heatmaps computed by passing the attention maps through a special softmax layer [57]. The second branch, called the attention enhancement branch, also computes a per-joint attention map but does so under no constraints, allowing it to freely learn to detect the hand pixels most important for estimating the joint depth values under different scenarios. This attention map enhances the attention map computed by the UV branch through a fusion operation, which is performed by a linear interpolation controlled by per-joint learnable parameters. As a result, the fused attention maps attend to not only the joint pixels but also the hand pixels that do not belong to joints but contain useful information for estimating the joint depth values. The fused attention map is then used as guidance for pooling features from the depth feature map computed by the depth branch. Finally, a weight-sharing linear layer is used to estimate the joint depth values from the feature vectors computed for each joint.

The feature pooling is conducted through a dot-product operation. This type of feature pooling followed by an estimation layer shared across all the joints is adopted from Transformer networks [82]. This approach breaks with the standard approach in hand pose estimation methods, where they use dedicated layers for estimating the location of each joint. We show in the sec 3.5.3 that TriHorn-Net derives its power from the proposed estimation formulation, namely the decomposition of the hand pose estimation into the estimation of 2D joint locations and the estimation of their corresponding depth values aided by two separate attention maps.

3.4.1 Encoder

The encoder is defined as a non-linear mapping from the input depth image to the output feature volume $f : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{c \times h \times w}$, where h, w and c denote the height, width and the number of the channels of the output feature volume respectively. While any off-the-shelf network architecture can be used as this non-linear mapping, we empirically find that, to maximize accuracy, the encoder network should have a high capability of extracting and fusing features at different scales. This is because the hand orientation, the arrangement of the fingers, and the relationships of adjacent joints are among the many cues that are best recognized at different scales in the depth image. We show in the sec 3.5.3 that the proposed model is robust to the choice of the encoder network architecture as long as the above-mentioned requirement of extracting and fusing features at different scales is met. We use an Hourglass network [83] as the encoder. It uses skip-connections and repeated bottom-up, top-down processing to extract and consolidate features across different scales.

3.4.2 UV Branch

This branch takes as input the output feature volume from the encoder and computes a per-joint attention map $Att_{uv} \in \mathbb{R}^{J \times h \times w}$. We use $Att^j_{uv} \in \mathbb{R}^{h \times w}$ to refer to the attention map corresponding to the j^{th} joint. Att^j_{uv} is explicitly encouraged to focus on pixels where the j^{th} joint occurs by applying 2D supervision to it. To this end, the attention map Att^j_{uv} is first normalized by a spatial softmax layer [57] to obtain the corresponding heatmap $H^{2D}_j = \sigma(Att^j_{uv})$ as follows:

$$H^{2D}_j(x, y) = \frac{exp(Att^j_{uv}(x, y))}{\sum\limits_{u_i, v_i \in \Omega} exp(Att^j_{uv}(u_i, v_i))} \tag{3.1}$$

In the above, σ denotes the spatial softmax layer. The heatmap H^{2D}_j represents the likelihood of the j^{th} joint occurring at each pixel location. Ω represents the spacial domain of the attention map Att^j_{uv}. The 2D location of the j^{th} joint is computed through an integration operation similar to [84, 57], as follows:

$$(\bar{U}^j, \bar{V}^j) = \sum_{u_i} \sum_{v_i} (u_i, v_i) H^{2D}_j(u_i, v_i) \tag{3.2}$$

In the above, $(\bar{U}^j, \bar{V}^j)$ represents the estimated coordinates of the j^{th} joint in the depth image space. The supervision is applied to all attention maps Att^j_{uv} for $j \in 1, 2, ..., J$ by minimizing the mean L1 distance defined as:

$$\ell_{uv} = \frac{1}{2J} \sum_{j=1}^{J} |\bar{U}^j - U^j| + |\bar{V}^j - V^j| \tag{3.3}$$

In the above, (U^j, V^j) represents the ground-truth 2D location of the j^{th} joint. Note that $(\bar{U}^j, \bar{V}^j)$ is also used to report the estimated coordinates of the j^{th} joint in the depth image space.

3.4.3 Attention Enhancement Branch

This branch is aimed at computing a more flexible attention map to enhance the attention maps computed by the UV branch Att_{uv} towards facilitating the estimation of the joint depth values. Specifically, it takes as input the output feature volume from the encoder and computes a per-joint attention map $Att_{enh} \in \mathbb{R}^{J\times h\times w}$. $Att^{j}_{enh} \in \mathbb{R}^{h\times w}$ denotes the attention map corresponding to the j^{th} joint. In contrast to Att_{uv} , no external constraint (supervision) is applied to this attention map, allowing it to freely learn which hand pixels are the most relevant ones for estimating the depth value for each joint under different scenarios.

3.4.4 Depth Branch

Contrary to the common practice of computing pixel-wise depth offset or prediction, the proposed model computes dense pixel-wise depth feature vectors. Specifically, this branch takes as input the output feature volume from the encoder and produces a dense depth feature map $\mathfrak{D} \in \mathbb{R}^{D\times h\times w}$. D represents the depth feature vector dimension, which is set $D = 64$ in our experiments. The depth feature vector at the special location (x, y) in the depth feature map $\mathfrak{D}$, denoted by $\mathfrak{D}(x, y) \in \mathbb{R}^{D}$, is developed such that it contains information about the depth value of the joints gathered from the input depth image pixels included in the receptive field of the special location (x, y) in the depth feature volume. The final feature vector used for each joint to estimate its depth value is obtained using a weighted average computed over all the depth feature vectors, where the weight to each depth feature vector is assigned using the corresponding fused attention map.

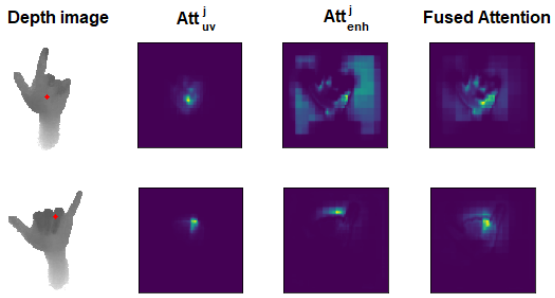

Figure 3.2: Qualitative examples of how the two attention maps complement each other to guide feature pooling for depth estimation. Each row shows an example including the input depth image, the Att^j_{uv} and Att^j_{enh} attention maps computed by the first and second branches respectively, and the resulting fused attention map Att^j_{fused}. The red dot in the input depth image marks the ground-truth joint location for which the attention maps are computed

3.4.5 Attention Fusion

The two attention maps Att^j_{uv} and Att^j_{enh} are complementary in the sense that Att^j_{uv} shifts the network attention to the pixels where the j^{th} joint occurs, while Att^j_{enh} helps the network pay attention to the non-joint pixels that might contain useful information for estimating the depth value of the j^{th} joint. These two attention maps are fused as follows:

$$Att^j_{fused} = \sigma(\beta_j Att^j_{uv} + (1 - \beta_j) Att^j_{enh}) \tag{3.4}$$

Here, $\beta_j \in [0, 1]$ denotes the learned parameter that controls the contribution of each attention map to the fused attention map Att^j_{fused}. The proposed model uses Att^j_{fused} as guidance to pool features from the pixels that contain the most relevant information with respect to the depth of the j^{th} joint. Fig. 3.2 shows some qualitative examples of how the two attention maps Att^j_{uv} and Att^j_{enh} play the complementary role in forming the final fused attention map Att^j_{fused} in order to guide the subsequent feature pooling for depth value estimation.

3.4.6 Depth Value Estimation

The depth value for the j^{th} joint is estimated from the feature vector obtained by pooling features from the pixels that contain the most relevant information about its depth, which is guided by Att^{j}_{fused} as follows:

$$F_j = Att^{j}_{fused} \circ \mathfrak{D} = \sum_{x} \sum_{y} Att^{j}_{fused}(x, y)\mathfrak{D}(x, y) \tag{3.5}$$

Where $F_j \in \mathbb{R}^D$ denotes the pooled feature vector for the j^{th} joint. The depth value for the j^{th} joint, denoted by $\bar{Z}_j$, is then estimated using a single linear layer as follows:

$$\bar{Z}_j = F_j \boldsymbol{W} + \boldsymbol{b} \tag{3.6}$$

Where $\boldsymbol{W} \in \mathbb{R}^D$ and $\boldsymbol{b} \in \mathbb{R}$ denote the weights of the linear layer. Note that this linear layer is shared across all the joints. This not only improves the parameter efficiency but also encourages ensemble-like behavior in the depth feature vectors, as each has to gather the depth information of all joints. This type of feature pooling using attention followed by a shared layer is inspired from the mechanism employed in Transformer networks [82].

The depth value estimation for the joints is supervised by the following loss term:

$$\ell_d = \frac{1}{J} \sum_{j=1}^{J} |\bar{Z}_j - Z_j| \tag{3.7}$$

Where Z_j refers to the ground-truth depth value for the j^{th} joint.

3.4.7 End-to-End Training

The proposed model is end-to-end differentiable and is trained by minimizing the loss function that comprises the two loss terms discussed in the previous sections, which is formulated as:

$$\mathcal{L} = \ell_{uv} + \lambda \ell_d \tag{3.8}$$

where λ is a weighting factor to balance ℓ_{uv} and ℓ_d. We set $\lambda = 1$ in our experiments.

3.4.8 PixDropout

The proposed data augmentation function PixDropout is defined as a function $\mathcal{T}_\alpha : \mathbb{R}^{H\times W} \rightarrow \mathbb{R}^{H\times W}$, where the parameter $\alpha \in [0, 1]$ controls the intensity of the augmentation. In the first step, we uniformly sample a probability γ from the range $[0, \alpha]$. In the second step, we uniformly sample a set of the pixels $\mathcal{Q}$ from the hand surface. Each pixel is selected with a probability of γ. The augmented depth image $\hat{D_I} = \mathcal{T}_\alpha(D_I)$ is computed by dropping the selected pixels as follows:

$$\hat{D_I}(p) = \begin{cases} C & \text{if } p \in \mathcal{Q} \\ D_I(p) & \text{otherwise} \end{cases} \tag{3.9}$$

Where p denotes an arbitrary pixel in the depth image. C represents the constant value assigned to the background (non-hand) pixels.

3.5 Experiments

3.5.1 Implementation Details

The pre-processing method for preparing the input depth image includes first cropping the hand area from a depth image similar to [85], and then resizing it to a fixed size of 128x128. The depth values are normalized to [-1, 1]. In order to maximize accuracy, the encoder's output feature volume needs to be of high spatial dimension. To strike a balance between the computational complexity and performance, we set it to be half of that of the input depth image. We use Adam [86] optimizer with a cosine learning rate decay schedule [87] for training. The initial learning rate and the weight decay are set to be 10^{-3} and 10^{-5} respectively. For data augmentation, we use geometric transformations including in-plane rotation ([-180, 180] degree), 3D scaling

Table 3.1: The performance of the models with and without (w/o) PixDropout as augmentation function on ICVL [1]. The numbers indicate the mean distance error (mm)

Model	with PixDropout	w/o PixDropout
TriHorn-Net	**5.73**	5.90
ResNet-50	**7.71**	7.81

Table 3.2: Impact of using different attention maps for depth feature pooling on the performance on ICVL [1]

Attention Map	Error (mm)
Att_{uv}	5.91
Att_{enh}	6.03
Fused Attention	**5.73**

([0.9, 1,1]), and 3D translation ([-8, 8] mm), as well as the proposed PixDropout as the appearance transformation. We set $\alpha = 0.15$ in all the experiments. We trained the model for 40 epochs on ICVL, 40 epochs on NYU and 60 epochs on MSRA. All experiments are implemented by PyTorch framework [88] and conducted on a single server with one NVIDIA 1080Ti GPU.

3.5.2 Datasets and Evaluation Metrics

ICVL Dataset. The ICVL dataset [1] provides 22K and 1.6K depth frames for training and testing, respectively. The ground-truth for each frame contains J = 16 joints, including one joint for the palm and three joints for each finger. We do not use the additional 300k augmented frames (which are obtained with in-plane rotations of the original training frames) included in this dataset.

MSRA Dataset. The MSRA dataset [3] contains more than 76K frames captured from 9 subjects. Each subject contains 17 hand gestures and each hand gesture has about 500 frames with segmented hand depth image. Each frame is provided with a ground-truth of J = 21 joints, including one joint for the wrist and

Table 3.3: Comparison of different choices for the encoder network architecture on ICVL [1]. #Params indicates the number of the model parameters

Encoder Architecture	Error (mm)	#Params
HRnet [89]	5.91	7.22M
ResDeconv [90]	6.04	27.24M
Hourglass [83]	**5.73**	7.81M

Table 3.4: Comparison of different attention fusion approaches on ICVL [1]

Approach	Error (mm)
Concatenation	5.98
Summation	5.84
Ours	**5.73**

four joints for each finger. Following the protocol used by [3], we evaluate the proposed method on this dataset with the leave-one-subject-out cross-validation strategy.

NYU Dataset. The NYU dataset [2] is captured from three different views with Microsoft Kinect sensor. Each view contains 72K training 8K testing depth images. Following the common protocol, we only use the first view with a subset of J = 14 joints out of total of 36 annotated joints provided for both the training and testing.

Evaluation metrics. We use the two most commonly used metrics for evaluation of 3D hand pose estimation: the mean distance error (in mm) and the success rate. The mean distance error measures the average Euclidean distance between the estimated and the ground-truth coordinates computed across all the joints and over the entire testing set. The success rate is defined as the fraction of the frames for which the mean distance error is less than a certain distance threshold.

3.5.3 Ablation Study

Impact of using complementary attention maps. We study the impact of employing two complementary attention maps in the proposed model. Specifically, we examine the performance of the model in three cases with respect to the attention map used for depth feature pooling. The first case only uses the attention map computed by the UV branch and removes the attention enhancement branch from the network. The second case only uses Att_{enh}, which is computed by the attention enhancement branch. The third case corresponds to the proposed approach based on fusing the two complementary attention maps. As can be seen in Table 3.2, using a second freely learned attention map enhancing the attention map computed by the UV branch leads to the best performing case.

Effectiveness of Different Approaches for Attention Fusion. We study the effectiveness of the proposed attention fusion approach in our model. We implement the proposed model using three different approaches for attention fusion: 1) concatenation 2) summation and 3) the proposed strategy. For concatenation, the two attention maps are first concatenated and then passed through a number of convolutional layers to obtain the fused attention map. For the second experiment, the two attention maps are simply fused by the element-wise addition. The proposed strategy is an extension of the summation approach, where the contribution of each attention map to the fused attention map is controlled by a learned parameter. As can be seen in Table 3.4, the proposed strategy performs best.

Impact of Different Encoder Network Architectures. We analyse the impact of different encoder network architectures on the model performance. Specifically, we use three representative architectures: 1) Hourglass network [83], 2) HRNet [89] and 3) ResNetDeconv [90]. Although through different mechanisms, Hourglass network and HRNet both directly transfer the low-level features extracted in the

early layers to deeper layers via direct skip-connections. On the other hand, ResNet-Deconv down-samples the input to a low resolution feature map and then up-samples it back by a deconvolution head. As can be seen in Table 3.3, the Hourglass network and HRNet both achieve the state-of-the-art results. Although ResNetDeconv does not have the same ability to extract and fuse features at different scales, it still performs strongly compared to the existing methods. These observations demonstrate that the proposed model is robust to the choice of the encoder network architecture and derives its superior performance from our novel formulation of pose estimation.

3.5.4 Effectiveness of PixDropout

To verify the effectiveness of the proposed data augmentation function PixDropout, we conduct two independent comparisons. We compare the performance of the proposed model with and without PixDropout. We also repeat this comparison using a regression-based method. Specifically, we use a ResNet-50 network [42], with its last fully connected layer replaced by 2 fully connected layers to estimate the hand pose. As can be seen in Table 3.1, PixDropout leads to a performance improvement not only in the proposed model but also in a model of different nature, demonstrating its effectiveness as an augmentation function for depth-based hand pose estimation methods.

3.5.5 Comparison with the State-of-the-Art Methods

We compare the proposed model with the state-of-the-art methods including both dense detection-based methods and regression-based methods. These methods include model-based method (DeepModel) [46], DeepPrior [45], improved DeepPrior (DeepPrior++) [85], region ensemble network (Ren-4x6x6 [91], Ren-9x6x6 [92]), Pose-Ren [94], Generalized-Feedback [95], dense regression network (DenseReg) [11], A2J

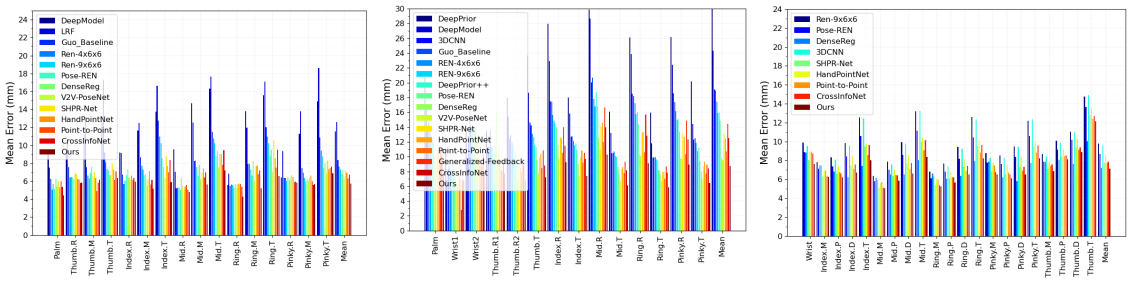

Figure 3.3: Comparison with the state-of-the-art methods on ICVL [1] (Left), NYU [2] (Middle), and MSRA [3] (Right) datasets. The per-joint mean error is used for comparison (R: root, T: tip).

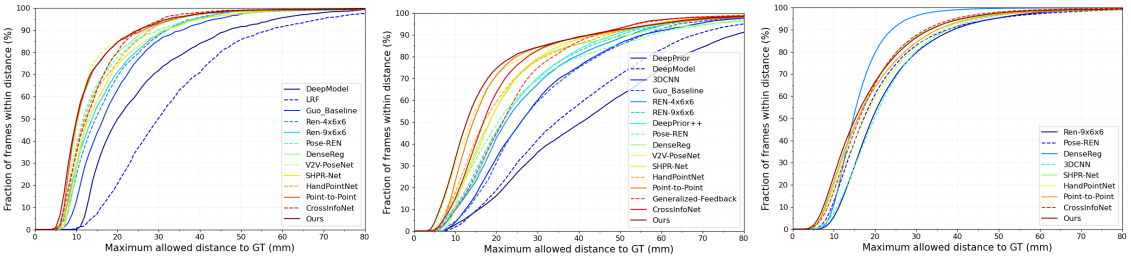

Figure 3.4: Comparison with the state-of-the-art methods on ICVL [1] (Left), NYU [2] (Middle), and MSRA [3] (Right) datasets. Success rates over different error thresholds is used for comparison.

[9], CrossInfoNet [8] and JGR-P2O [56], 3DCNN [49], SHPR-Net [93], HandPointNet [51], Point-to-Point [10], NARHT [73], HandFoldingNet [58] and V2V [52]. Fig. 3.4 and Fig. 3.3 respectively show the success rate and per-joint mean error (mm) on the ICVL, NYU and MSRA datasets. Table 3.5 summarizes the performance based on the mean distance error on the three datasets. The results show that the proposed method significantly outperforms the state-of-the-art methods on all of these three benchmark datasets, achieving a mean distance error of 5.73 mm, 7.68 mm and 7.13 mm on ICVL, NYU and MSRA respectively.

Table 3.5: Comparison with the state-of-the-art methods on ICVL [1] (Left), NYU [2] (Middle), and MSRA [3] (Right). "Error" indicates the mean distance error in (mm)

Methods	Error
DeepModel [46]	11.56
DeepPrior [45]	10.4
DeepPrior++ [85]	8.1
REN-4x6x6 [91]	7.63
REN-9x6x6 [92]	7.31
DenseReg [11]	7.3
SHPR-Net [93]	7.22
HandPointNet [51]	6.94
Pose-REN [94]	6.79
CrossInfoNet [8]	6.73
NARHT [73]	6.47
A2J [9]	6.46
Point-to-Point [10]	6.3
V2V-PoseNet [52]	6.28
JGR-P2O [56]	6.02
HandFoldingNet [58]	5.95
Ours	**5.73**

Methods	Error
DeepPrior [45]	19.73
DeepModel [46]	17.04
3DCNN [49]	14.1
REN-4x6x6 [91]	13.39
REN-9x6x6 [92]	12.69
DeepPrior++ [85]	12.24
Pose-REN [94]	11.81
Generalized-Feedback [95]	10.89
SHPR-Net [93]	10.78
HandPointNet [51]	10.54
DenseReg [11]	10.2
CrossInfoNet [8]	10.08
NARHT [73]	9.8
Point-to-Point [10]	9.1
A2J [9]	8.61
HandFoldingNet [58]	8.58
V2V-PoseNet [52]	8.42
JGR-P2O [56]	8.29
Ours	**7.68**

Methods	Error
REN-9x6x6 [92]	9.79
3DCNN [49]	9.58
DeepPrior++ [85]	9.5
Pose-REN [94]	8.65
HandPointNet [51]	8.5
CrossInfoNet [8]	7.86
SHPR-Net [93]	7.76
Point-to-Point [10]	7.7
V2V-PoseNet [52]	7.59
JGR-P2O [56]	7.55
NARHT [73]	7.55
HandFoldingNet [58]	7.34
DenseReg [11]	7.23
Ours	**7.13**

CHAPTER 4

Semi-supervised 3D Hand Pose Estimation

4.1 Introduction

The availability of more accurate and affordable commodity depth cameras coupled with the success of Deep Neural Networks (DNN) has led to significant progress in depth-based 3D hand pose estimation in the recent years [5, 6, 7, 8, 9, 10, 11, 12, 2]. Despite these advancements, one major challenge that remains is that DNN-based methods require large amounts of annotated training data to realize their full potential. However, collecting such data is both costly and time-consuming. In this chapter, we propose a semi-supervised 3D hand pose estimation method to significantly reduce the dependence on labeled training data.

A straightforward approach to mitigate the requirement for labeled training data is to use synthesized training data with accurate annotations [4], which can be generated with minimal human effort. However, models trained on synthesized data generalize poorly to the real-world data due to the significant domain gap between synthetic and real-world data. A popular alternative is semi-supervised learning (SSL) [96], where the goal is to leverage unlabeled data along with the labeled data, hence reducing the amount of labeled data required for training. Most of the recent advancements of SSL methods have been focused on image classification [97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107].

Semi-supervised learning has recently attracted attention in the area of 3D hand pose estimation. Chen et al. [7] leverage unlabeled data by minimizing the Chamfer loss between the input point cloud and its reconstructed version by a decoder. Wan

et al. [108] jointly train two deep generative models with a shared latent space to model the statistical relationships of depth images and their corresponding hand poses. Their architectural design facilitates learning from unlabeled data. Poier et al. [4] exploit synthetic data to reduce reliance on annotated real-world data by learning to map from the features of real data to that of synthetic data. Baek et al. [12] synthesize data in the skeleton space and then its corresponding depth map to augment the training data. These methods enable semi-supervised learning through accommodations in their network architecture. Orthogonal to this line of work, we propose a model-agnostic semi-supervised framework for 3D hand pose estimation that takes advantage of the most recent advancements of SSL methods in image classification.

The proposed framework consists of two identical networks that are trained jointly: 1) student network and 2) teacher network. Any off-the-shelf network architecture can be used as long as it provides a means for prediction uncertainty estimation. For training the teacher, we adopt an approach based on consistency training [109]. Driven by the intuition that a good model should be robust to any small change in an input example, approaches based on the consistency training enforce the model predictions to be invariant to small noise applied to input examples. Inspired by this approach, we train the teacher network using both the labeled and unlabeled parts of the training data, with a combination of the typical supervised loss and an unsupervised loss formulated in such a way to enforce model consistency defined as the model output equivariance under a set of affine transformations.

Note that the proposed method uses different training strategies from [109] due to the fundamentally different nature of the 3D hand pose estimation, which is a structured regression task as opposed to an image classification task. This difference poses unique challenges for a hand pose estimation method based on consistency training,

necessitating not only architectural changes but also different training strategies. We present several novel components to effectively address these challenges. The student network is trained using the pseudo-labels generated by the teacher network. More specifically, to stabilize the training, exponential moving average [110] of the teacher network's parameters are used for generating the pseudo-labels. After the training is finished, the student network is fine-tuned on the labeled part of the training data since it has not seen any of them during training. Note that the proposed method comes at no additional cost at test time, as only the student network is used for inference and the teacher network is discarded after training.

It should be stressed that the proposed training of a separate student network is different from knowledge distillation [111]. The goal of knowledge distillation is to transfer the knowledge of a complicated model to a simpler model by training the simpler model with the softmax outputs of the complicated model. Moreover, knowledge distillation is performed after training. However, the proposed method employs two identical networks that are trained simultaneously. Furthermore, the student is trained using the exponential moving average (EMAN) of the teacher network's parameters.

We conduct an extensive evaluation of the proposed method on three publicly available datasets, namely ICVL [1], MSRA [3] and NYU [2], which are challenging benchmarks commonly used for evaluation of 3D hand pose estimation methods. The results demonstrate that the proposed method significantly outperforms the current state-of-the-art semi-supervised hand pose estimation methods. We also analyse the performance of the proposed method in cases of severe scarcity of ground-truth annotations and show its effectiveness under such scenarios. Most remarkably, using only 25% of the annotations, the proposed method preforms on par with the state-

of-the-art fully supervised methods (methods that use 100% of the ground-truth annotations).

In summary, the contributions in this chapter are as follows:

- We propose a novel semi-supervised hand pose estimation method to effectively leverage the unlabeled data. To the best of our knowledge, this is the first method to incorporate consistency training for semi-supervised training on depth images of hands.
- We propose several novel strategies to enable consistency training for 3D hand pose estimation on depth images.
- The proposed method is the first depth-based hand pose estimation method to incorporate advances from recent SSL methods such as [112, 109, 110], which target general-purpose image classification. A key contribution is proposing concrete ways to apply those ideas to depth-based hand pose estimation, and showing that they lead to improved performance.
- We empirically show that the proposed method outperforms the current state-of-the-art semi-supervised 3D hand pose estimation methods.

4.2 Hand Pose Estimation

Hand pose estimation has been a long-standing problem in the Computer Vision community. While early methods relied on non-data-driven approaches such as hand crafted features, optimization methods, and distance metrics [59, 32, 35], in recent years there has been a shift to methods based on deep neural networks (DNNs). Oberweger et al. [45] proposed a method to estimate the hand pose represented by PCA coefficients of a statistical hand model. Wang et al. [92] use the ensemble principle by partitioning the last convolutional outputs of a CNN into several regions and using separate regressors to estimate the hand joints. Another line of work is to

take advantage of the recent advancements in 3D Deep learning. To this end, [49, 52] converted 2.5D depth images into 3D voxels and employed 3D CNNs to estimate the 3D hand pose. Several methods [51, 10, 54, 7, 58, 73] have been recently proposed to utilize point cloud processing networks by converting depth images into point clouds as the input. Comprehensive reviews of depth-based hand pose estimation can be found in [113, 5].

The above-mentioned methods are all fully-supervised. Our work is most related to the recent line of semi-supervised methods for 3D hand pose estimation [7, 108, 4, 12]. SemiHand [114] is closest to our method. It uses consistency training, which is also the case for the proposed method. However, the consistency training approach in the proposed method is significantly different than that of [114]. Besides using a different input data modality (2.5 depth images as opposed to RGB images), the proposed framework is based on a student-teacher paradigm and uses consistency training only for training the teacher network, whereas SemiHand [114] uses a single network. The proposed framework also uses a fundamentally different mechanism for label-refinement and the network uncertainty estimation. Finally, unlike SemiHand [114], the proposed method only uses view consistency.

4.3 Semi-supervised Learning in Image Classification

The key challenge to training of modern DNNs is the requirement for large amounts of labeled data. Semi-supervised learning (SSL) mitigates this requirement by providing a means of leveraging unlabeled data. Classic examples of SSL methods include transductive models [97, 98, 99], entropy minimization [100], co-training [101, 102] and graph-based models [103, 104, 105, 106, 107].

Our work is closely related to the recent line of SSL methods based on pseudo-labeling [115, 116, 117], where they produce artificial label for unlabeled data samples

and train the model to predict the artificial label when fed unlabeled samples as input, and consistency training [112, 109, 118] wherein they enforce the model predictions to be consistent across a sample and its perturbed version. However, the proposed method is fundamentally different from the methods discussed above. They are all focused on image classification, where the goal is encourage representation invariance across different views of the same image. However, the proposed method performs hand pose estimation, which is a structured regression task. It critically depends on spatial information and its goal is to enforce representation equivariance across different views. These differences pose unique challenges for a hand pose estimation method based on consistency training. In this chapter, we propose several novel strategies to address these challenges and take advantage of the state-of-the-art SSL methods in image classification.

4.4 Proposed Method

4.4.1 Problem Formulation and Notation

The task of 3D hand pose estimation is defined as follows: given an input depth image $x \in \mathbb{R}^{H \times W}$, the task is to estimate the 3D location of a set of pre-defined hand joints $\mathcal{J} \in \mathbb{R}^{N_J \times 3}$ in the camera coordinate system by learning a mapping f in the form of a neural network parameterized by θ, such that $\mathcal{J} = f(X; \theta)$. H and W denote the height and width of the depth map respectively. For the sake of simplicity, we refer to the input data dimensionality as $d = H \times W$. We use N_J to refer to the number of estimated joints. $\mathcal{J}_i = (U, V, Z)$ represents the location of the the i^{th} joint .The function f is learned using the training set consisting of labeled examples $(x_l, \mathcal{J}^l) \sim P_L$ and unlabeled examples $x_u \sim P_U$. P_l and P_U denote the probability distributions of labeled and unlabeled examples respectively. We define an

augmentation function $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that it maintains the equivariance property. This mathematically means that if $x' = \Phi(x)$, then we have $\mathcal{J}' = \Phi(\mathcal{J})$. We define $\mathcal{M}$ as a uniform probability distribution over all such augmentation functions. In our experiments, we use a subset of affine transformations including translation, scaling and rotation as augmentation functions.

The overview of the proposed method is illustrated in Fig. 4.1. It employs two identical networks called the teacher network and the student network, whose parameters are denoted by θ_T and θ_S respectively. The teacher network task is to provide supervisory signal for the student network by generating pseudo-labels. It is trained using a combination of the typical supervised loss and consistency training loss. As the teacher network improves, so do the pseudo-labels it generates for the student network. As a result, the student network keeps improving as the training of the teacher network progresses. After training is finished, the student network will be fine-tuned using the available labeled samples because it has not seen any of them during training. We empirically found that this leads to some modest performance improvements.

4.4.2 Network Architecture

Both the teacher and student network follow the same architecture that is similar to [57]. It consists of an encoder network and two separate branches. The encoder is a CNN whose task is to extract hand features from the input depth image. Its output feature volume serves as the input to two branches. The first branch estimates a heatmap H_j^{2D} for each joint. The 2D heatmap H_j^{2D} represents the occurrence likelihood of the j^{th} joint at each pixel location. The second branch estimates a depth map H_j^z for each joint. H_j^z represents depth prediction for the corresponding pixels for the j^{th} joint. The 3D locations are then computed following [84, 57]:

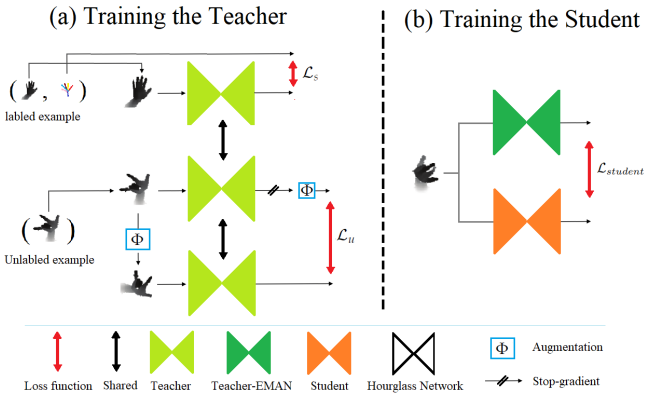

Figure 4.1: Left: overview of training the teacher network. It depicts a training batch consisting of one labeled example and one unlabeled example. The teacher network is trained using a combination of the typical supervised loss on the labeled examples and consistency loss on the unlabeled examples. Right: the student network is trained using the pseudo-labels provided by the EMAN of the teacher network's parameters

$$(U^j, V^j) = \sum_{u_i} \sum_{v_i} (u_i, v_i) \hat{H}_j^{2D}(u_i, v_i) \tag{4.1}$$

$$Z^j = \sum_{u_i} \sum_{v_i} H_j^z(u_i, v_i) \hat{H}_j^{2D}(u_i, v_i) \tag{4.2}$$

In the above, $\hat{H}_j^{2D}(u_i, v_i)$ is the H_j^{2D} normalized through spatial Softmax operation as follows:

$$\hat{H}_j^{2D}(x, y) = \frac{exp(\alpha_j H_j^{2D}(x, y))}{\sum_{u_i, v_i \in \Omega} exp(\alpha_j H_j^{2D}(u_i, v_i))} \tag{4.3}$$

Here, Ω represents the set of all pixel locations in the input map H_j^{2D}. α_j denotes the temperature parameter that controls the spread of the output heatmaps $\hat{H}_j^{2D}$. Unlike [57] that trains these parameters along with the rest of the network parameters, we set $\alpha_j = 1$ for $j \in 1, 2, ..., N_J$ and keep them fixed during the training phase. In our experiments, we use an Hourglass network [83] as the encoder.

4.4.3 Teacher Network Training

The teacher network is trained using both labeled and unlabeled examples by solving the following optimization problem, conceptually similar to [109, 114]:

$$\begin{aligned}\min_{\theta_T} \mathcal{L}_s(\theta_T) + \lambda \mathcal{L}_u(\theta_T) =& \mathbb{E}_{(x_l, \mathcal{J}^l) \sim P_L(x)}[D(f(x_l; \theta_T), \mathcal{J}^l)] + \\ & \lambda \mathbb{E}_{x_u \sim P_U(x)} \mathbb{E}_{\Phi \sim \mathcal{M}}[D(\Phi(f(x_u; \bar{\theta_T})), f(\Phi(x_u); \theta_T))]\end{aligned} \tag{4.4}$$

Here, D denotes mean element-wise L1 distance. $\bar{\theta_T}$ denotes a fixed copy of the current parameters θ_T, indicating the the gradient is not back-propagated through $\bar{\theta_T}$, as done in [109, 119]. λ is a weighting factor to balance the terms $\mathcal{L}_s$ and $\mathcal{L}_u$. $\mathcal{L}_s$ is the typical supervised loss computed on the labeled examples, which is aimed at minimizing the difference between the model predictions and the corresponding ground-truth labels. $\mathcal{L}_u$ is the unsupervised consistency regularization loss computed on unlabeled examples to enforce consistency of the model predictions across different views of the same depth images. In contrast to image classification where the consistency is defined as the model prediction invariance across different views [109], we define consistency as equivariance under a set of affine transformations, similar to SemiHand [114]. Explicitly enforcing such an estimation equivariance on the unlabeled examples proves to be a very effective means of leveraging them. $f(x_u; \bar{\theta_T})$ can be interpreted as the pseudo-labels generated by the teacher network from the view x_u to be used by its own in the second view $\Phi(x_u)$.

Sample Masking. Models trained using self-generated pseudo-labels generally suffer from confirmation bias [120], where the model keeps amplifying its own errors. It has been demonstrated that masking out noisy pseudo-labels and maintaining only high-quality ones for training can considerably reduce the confirmation bias [115]. The standard approach towards this end has been to base the decision of whether to use

a pseudo-label for training on a model prediction certainty(or uncertainty) measure compared against a threshold. The typical approach in image classification, namely taking the maximum of the model output probability, is not applicable to the proposed method due to the different nature of its task. SemiHand [114] defined the model confidence on a data sample as the sum of the distance between the model's prediction on an image and that of its randomly perturbed version, and the distance between the pseudo-label and its corrected pseudo-label. However, this approach requires additional model evaluations and performing forward kinematic chain, which adds computational overhead. The proposed method uses a simple yet effective method to measure the model uncertainty. The prediction uncertainty for the j^{th} joint, denoted by C_j, is approximated using the Standard deviation (STD) of the corresponding estimated normalized heatmap $\hat{H}_j^{2D}$, computed as follows:

$$C_j = \sqrt{\sum_{u_i} \sum_{v_i} \hat{H}_j^{2D}(u_i, v_i) \left\| [U_j, V_j]^T - [u_i, v_i]^T \right\|^2} \tag{4.5}$$

Here, $\|.\|$ denotes the Frobenius norm function. We empirically found that when the model is certain about its prediction on a given joint, its corresponding heatmap has a low STD. On the other hand, when the model is not certain and there are many candidate pixels, the heatmap tends to be wider (and hence high STD).

We define the mask $M \in \mathbb{R}^J$ as follows:

$$M_j = \begin{cases} m_a & \text{if } C_j < T_j \\ m_r & \text{otherwise} \end{cases} \tag{4.6}$$

Here, T_j is the threshold used for masking the j^{th} joint. Symbols m_a and m_r denote the weights given to the pseudo-labels that are respectively accepted and rejected. Image classification SSL methods such as [112, 109, 121] have traditionally

taken a binary approach for masking ($m_a = 1$ and $m_r = 0$), which is a special case of Eq. 4.6. While this binary approach has proven effective for image classification, we empirically found that including the pseudo-labels that are rejected in the training ($m_r \neq 0$) consistently leads to performance improvement in the proposed method. The L1 distance D in $\mathcal{L}_u$ in the Eq. 4.4 is replaced by the following weighted average:

$$D(\mathcal{J}, \mathcal{J}') = \frac{1}{3K} \sum_{j=1}^{N_J} M_j \sum_{i=1}^{3} |\mathcal{J}_{ji} - \mathcal{J}'_{ji}| \tag{4.7}$$

where $K = \sum_j M_j$. In the case where masking is not used, we set all $M_j = 1$.

Dynamic Thresholding. A standard practice of SSL methods in image classification is to use a fixed threshold for masking [112, 109]. Most recently, [121] employed a strategy for dynamic adjustment of thresholds for different classes based on class learning effects. However, none of the these strategies is practical for the proposed method. Naively using fixed thresholds for masking causes two issues in the proposed method. First, since the the uncertainty measure for different joints are of different scales (e.g. heatmaps corresponding to fingertips are usually very peaky but are relatively wide for the palm, leading to low and high STDs respectively), we would need N_J different thresholds (one for each joint), making the hyper-parameter optimization complicated. Secondly, adopting uncertainty-based pseudo-labeling leads to a class imbalance in the pseudo-labels, and thereby, misguides the training [122].

To tackle these issues, we employ a strategy to dynamically adjust the thresholds. Let ρ_t be the fraction of pseudo-labels allowed for training at the training epoch t. Let T_j^t be the threshold value used for masking for the j^{th} joint at the training epoch t. After initialization in the first epoch, T_j^{t+1} for the epoch $t+1$ is computed as follows:

$$T_j^{t+1} = T_j^t + \eta(\rho_t - \rho_t^j) \tag{4.8}$$

Here, ρ_t^j is the fraction of pseudo-labels corresponding to the j^{th} joint accepted for training in the epoch t according to the corresponding threshold T_j^t. Symbol η denotes the adjustment rate. Intuitively, when $\rho_t > \rho_t^j$, it means that the threshold T_j^t should increase to let pass more pseudo-labels. On the other hand, when $\rho_t < \rho_t^j$, it means the threshold T_j^t should decrease to accept fewer pseudo-labels for training. This type of addressing class-imbalance problem can be thought of as equivalent to Mean Sampling [123]. It ensures that a roughly equal fraction ρ_t of pseudo-labels for each joint is used for training in each epoch. We use a cosine schedule strategy [87] to increase ρ_t from the initial value ρ_{start} to reach its final value ρ_{end} over the course of training as follows:

$$\rho_t = \rho_{start} + 0.5(1 - \cos(\frac{T_{cur}}{T_{max}}\pi))(\rho_{end} - \rho_{start}) \tag{4.9}$$

Here, T_{cur} and T_{max} denote the current epoch number and the total number of training epochs respectively. Intuitively, the proposed method allows only a small proportion of pseudo-labels to be used at early phases of the training since the teacher network is still not accurate in early training phases. As the training progresses and the teacher network performance improves, a larger proportion of pseudo-labels are allowed to be used for training.

Dynamic Adjustment of the Training Signal Composition for the Teacher for Low-data Regime As discussed previously, the teacher network is trained by minimizing the Eq. 4.4, which is a combination of the typical supervised loss $\mathcal{L}_s$ and the unsupervised loss $\mathcal{L}_u$. A common scenario in a semi-supervised setting is when the access to labeled examples is very limited. In such scenarios, the inaccurate pseudo-labels generated by the teacher network in the early phases of training take a large number of training epochs to become sufficiently accurate for training. However, using inaccurate pseudo-labels for training misguides it and hinders con-

Table 4.1: The performance when using different strategies for adjusting λ. Cases where $\lambda_{start} = \lambda_{end}$ refer to the ones where λ is kept fixed during training

λ_{start}	λ_{end}	Mean Error (mm)
0.2	0.2	8.86
0.5	0.5	8.83
1.00	1.00	9.18
2.00	2.00	9.45
0.2	1.2	**8.71**

vergence. To tackle this issue, we dynamically adjust λ during training. To this end, it is initialized with λ_{start}, and then increased according to a exponential schedule to reach its final value at the last training epoch λ_{end}, as follows:

$$\lambda_t = \lambda_{start} + exp((\frac{T_{cur}}{T_{max}} - 1) * 5)(\lambda_{end} - \lambda_{start}) \tag{4.10}$$

Here, T_{cur} and T_{max} denote the current epoch number and the total number of training epochs respectively. The symbol λ_t represents the value of λ at the training epoch t. As can be seen in Table 4.1, the best performing case is when we give pseudo-labels a low weight when they are inaccurate in the beginning (e.g. $\lambda = 0.2$) and then gradually increase that weight as they become more accurate. Note that this approach also prevents the network from over-fitting to the small amount of the labeled data.

4.4.4 Student Network Training

The proposed method trains the student network using the unlabeled samples and their corresponding pseudo-labels generated by the teacher network. However, using pseudo-labels generated directly by the teacher network can lead to a potential problem. The teacher itself is constantly updated in the training, which could cause performance degradation and training instability in the student network since it has to learn to approximate a highly non-stationary function. An alternative is to

use the exponential moving average (EMA) of the teacher network's parameters to generate pseudo-labels [120]. However, this approach could lead to a potential mismatch between the EMA parameters and the batch normalization (BN) statistics in the parameter space [110] because the EMA parameters are averaged from the previous iterations, but the batch-wise BN statistics are instantly collected at the current iteration. We use the recently proposed fix for this issue called EMAN [110], where the batch-wise statistics are exponentially averaged from the previous iterations as well. The student network is trained by solving the following optimization problem:

$$\min_{\theta_S} \mathcal{L}_{student} = \mathbb{E}_{x_u \sim P_U(x)} \mathbb{E}_{\Phi \sim \mathcal{M}} [D(f(\Phi(x_u); \theta_{EMAN}), f(\Phi(x_u); \theta_S))] \tag{4.11}$$

where θ_{EMAN} denotes the exponentially moving averaged of the teacher network's parameters from the previous iterations computed as in [110]. $f(\Phi(x_u); \theta_{EMAN})$ are the pseudo-labels corresponding to x_u generated using EMAN parameters.

4.4.5 Training Algorithm

In summary, the proposed algorithm employs two identical networks that are trained simultaneously. The teacher network is trained using both the available labeled examples and the unlabeled examples, while the student network is trained using the unlabeled examples with their corresponding pseudo-labels generated by the teacher network. For training the teacher network, at each training iteration, we compute the supervised loss on a mini-batch of size B of labeled examples and compute the consistency loss on a mini-batch of size μB of unlabeled examples. We control the ratio of unlabeled examples batch size to the labeled examples batch size by μ. The pseudo code can be found in Algorithm 1.

Table 4.2: Hyper-parameters for the proposed method under different scenarios with respect to the percentage of available labeled examples for training

Hyper-parameter	% of Labeled Examples			
	1	25	50	75
μ	2	2	2	2
Batch size	16	16	16	16
η	0.8	0.8	0.8	0.8
λ_{start}	0.2	0.8	0.4	0.4
λ_{end}	1.2	1	0.6	0.6
ρ_{start}	0.2	0.4	0.4	0.4
ρ_{end}	0.8	0.8	0.8	0.8
m_a	1	1	1	1
m_r	0.25	0.25	0.25	0.25
T_{max}	1200	132	100	80

4.5 Experiments

4.5.1 Implementation Details

The input to the networks is prepared by cropping the hand area from a depth image following [85] and resizing it to a fixed size of 128x128. The depth values are then normalized to [-1, 1] for the cropped image. For training, we use Adam [86] optimizer with a cosine learning rate decay schedule [87]. The initial learning rate is set to be 10^{-4}, and a weight decay of 10^{-5} is used. The augmentation set used for $\mathcal{M}$ includes in-plane rotation ([-180, 180] degree), 3D scaling ([0.9, 1,1]), and 3D translation ([-10, 10] mm). We use PyTorch framework [88] for implementation. The proposed method uses a different combination of hyper-parameters depending on the number of labeled examples available for training. The combination of hyper-parameters for four different scenarios can be found in Table 4.2.

4.5.2 Datasets and Evaluation Metrics

We evaluate the proposed method on three public 3D hand pose estimation datasets: ICVL dataset [1], NYU dataset [2] and MSRA dataset [3]. The ICVL

Table 4.3: Performance under different strategies for adjusting the thresholds

Strategy	Error (mm)
Fixed Thresholds	10.43
$\rho_t = 0.4$	9.00
$\rho_t = 0.6$	8.84
$\rho_t = 0.8$	8.99
Ours	**8.71**

dataset contains 22K training and 1.5K testing depth images that are captured with an Intel Realsense camera. The ground truth hand pose of each image consists of $N_j = 16$ joints. The NYU dataset is captured with three Microsoft Kinects from different views. Each view consists of 72K training and 8K testing depth images. Following most previous works, we only use the frontal view and use a subset $N_j = 14$ out of the total 36 annotated joints for training and testing in all experiments. The MSRA dataset [3] contains more than 76K frames captured from 9 subjects. Each subject contains 17 hand gestures and each hand gesture has about 500 frames. Each frame is provided with a ground-truth of $N_J = 21$ joints. Following the protocol used by [3], we evaluate the proposed method on this dataset with the leave-one-subject-out cross-validation strategy.

For evaluation, we use one of the most commonly used metrics for evaluating 3D hand pose estimation methods: the mean distance error (measured in mm). The mean distance error represents the average Euclidean distance between the estimated and the ground-truth joint locations computed over the entire testing set.

4.5.3 Ablation Study

To conduct ablation study, we choose a scenario where labeled data is scarce (more specifically, 1% of the ground-truth labels are used) because such a scenario

Table 4.4: Impact of different masking approaches on the performance

Masking Approach	Error (mm)
No masking	9.79
Binary-masking	9.12
Ours	**8.71**

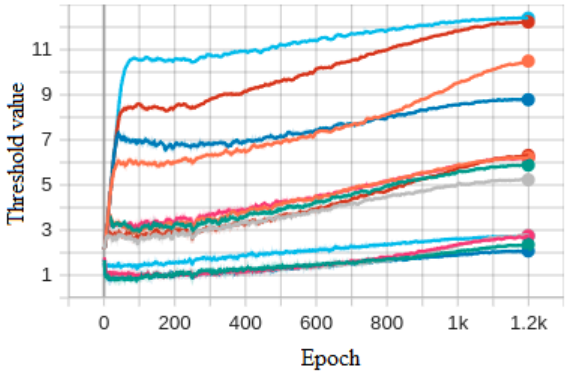

Figure 4.2: Example of how the proposed method systematically computes and then adjusts the thresholds for different joints as the training evolves. Each curve corresponds to a different joint

most highlights the proposed method's capability to leverage the unlabeled data. We use the NYU dataset for performing ablation study.

Impact of Sample Masking. We study the impact of using sample masking in the training. First, we use the ground-truth labels of the unlabeled data to illustrate how the accuracy of pseudo-labels improves when we use heatmap STDs as the uncertainty measure to mask out noisy pseudo-labels. As shown in Fig. 4.4, this approach leads to a consistent improvement of between 10% to 25% in the accuracy of pseudo-labels. As can be seen in Table 4.4, incorporating the proposed sample masking significantly improves the performance.

Effectiveness of Dynamic Thresholding. We examine the effectiveness of the the proposed dynamic thresholding strategy in our framework. We report the performance in three cases. In the first case, the thresholds are initialized and then

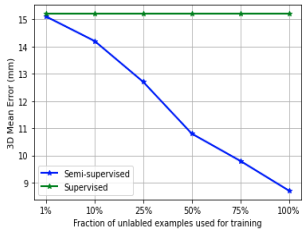

Figure 4.3: The performance of the model under different percentages of unlabeled examples used for training

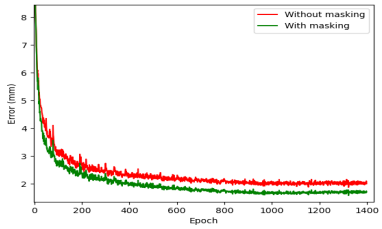

Figure 4.4: The accuracy of pseudo-labels generated by the teacher network with and without applying masking

kept fixed during training. The second case includes scenarios where the thresholds are dynamically adjusted according to Eq. 4.8, but ρ_t is kept fixed. The third case refers to the proposed strategy, where ρ_t is adjusted according to a cosine schedule [87]. As can be seen in Table 4.3, the proposed strategy leads to the best performing case.

We also shed more light on why the proposed dynamic thresholding strategy is a crucial component in the proposed framework. As discussed before, the uncertainty measure (heatmap STD) for different joints are of different scales. This would require N_J different thresholds, rendering the hyper-parameter optimization cumbersome. To tackle this issue, the proposed dynamic strategy systematically computes and then adjusts the thresholds for different joints in such a way that it roughly maintains the class-balance in the pseudo-labels accepted for training. An example of how the proposed strategy computes and adjusts different thresholds can be seen in Fig. 4.2.

Impact of Using a Separate Network as the Student. Empirical evidence demonstrate that using a separate network as the student (as opposed to using a single network as both the teacher and the student) leads to a performance improvement of 0.59 mm. We conjecture that this improvement primarily stems from the fact the the student network is trained using a more stationary data distribution (only the pairs

Table 4.5: Parameters used for generating pseudo-labels for training the student

Model Parameters	Error (mm)
Teacher	9.27
EMA-Teacher	10.28
EMAN-Teacher	**8.71**

of unlabeled examples and their pseudo-labels provided by the EMAN of the teacher network's parameters), as opposed to the teacher network that is trained using a combination of distributions (pairs of labeled examples and their ground-truth labels and pairs of unlabeled examples and their corresponding pseudo-labels).

Impact of Using EMAN for Generating Pseudo-Labels. To analyse the impact of using EMAN of the teacher network's parameters (instead of the parameters themselves) for generating the training pseudo-labels for the student network, we compare the performance of the student network in three cases in terms of the parameters used for generating pseudo-labels for its training: 1) the teacher network's parameters, 2) EMA-teacher [120] and 3) EMAN-teacher [110]. As can be seen in Table 4.5, the best performing case is when we use EMAN. EMAN greatly improves the stability of learning by constraining the target values for the student network to change more slowly.

Effectiveness of Post-training Fine-tuning of the Student Network. To verify the effectiveness of fine-tuning after the training is finished, we compare the performance of the student network with and without fine-tuning. As can be seen in Table 4.6, fine-tuning on the available labeled examples leads to further performance improvement in the proposed method. This is because the student network is only trained using the unlabeled examples and their corresponding pseudo-labels, and it has not seen any labeled examples during training.

Table 4.6: Student network performance with and without post-training fine-tuning on the available labeled examples

Fine-tuning	Mean Error (mm)
✗	8.80
✓	**8.71**

4.5.4 Ability To Leverage Unlabeled Examples

We analyze the effectiveness of the proposed method in leveraging unlabeled data, which is the key ability a semi-supervised method is aimed at achieving. Specifically, we use 1% of the dataset as the labeled portion of the training data, and gradually expand the unlabeled portion of the training data. As can be seen in Fig. 4.3, as the number of unlabeled data samples increases, the performance consistently improves. This clearly demonstrates the high capability of the proposed method of leveraging the unlabeled examples to improve its performance.

4.5.5 Comparison with State-of-the-Art Semi-Supervised Hand Pose Estimation Methods

To demonstrate the effectiveness of the proposed method, we compare it with the state-of-the-art depth-based semi-supervised methods including [7, 108, 12, 124]. Note that Beak et al. [12] particularly adopt a different approach from the rest of the works. They synthesize data in the skeleton space and train a separate network to synthesize its corresponding depth image. Although they use 100% of training data annotations, we include their work in our comparison as it is aimed at the same goal as the rest of the works. As can be seen in Table 4.7, the proposed method significantly outperforms the state-of-the-art methods. Most notably, the proposed method surpasses all the existing methods when only using 1% of ground-truth annotations for training.

The results from Table 4.7 also show that the proposed method enjoys a high label efficiency. Specifically, the performance of the proposed method reaches its highest level when using only 25% of the ground-truth annotations. It does not achieve a considerable performance gain when more ground-truth annotations are used. Interestingly, the performance gap between the cases where 1% and 100% of the ground-truth annotations are used is only 0.7 mm. These observations clearly demonstrate the high capability of the proposed method of taking full advantage of the unlabeled examples. On the other hand, the existing methods rely more on the labeled data. For example, for SO-HandNet [7], the performance gap between the cases where 25% and 100% of the ground-truth annotations are used is 3.7 mm and 3.4 mm on NYU and ICVL respectively, indicating a very lower label efficiency.

4.5.6 Semi-Supervised Learning under Extremely Low Data Regimes

We compare the proposed method with MURAUER [4], which to the best of our knowledge is the only depth-based 3D hand pose estimation method examined under scenarios of severe labeled data scarcity. As can be seen in Table 4.8, the proposed method significantly outperforms MURAUER [4] under the three out of the four scenarios. The only scenario where the proposed method is inferior is when there are only 10 labeled examples. This is because such a small amount of labeled training data lacks adequate information about the nature of the task. MURAUER [4] compensates for this lack of information by pre-training on synthetic data. However, the proposed method does not use any pre-training. More importantly, in contrast to MURAUER [4], the proposed method achieves this performance without the application-limiting requirement for multi-view real-data. Remarkably, in the case where there are only 100 labeled samples, the proposed method achieves 12.11 mm, which clearly demon-

Table 4.7: Comparison of the proposed method with state-of-the-art semi-supervised methods on ICVL and NYU datasets. The performance is evaluated by the test estimation error under different percentages of labeled data used for model training

Method	Label Usage	Augmented Set	ICVL(mm)	NYU(mm)
Beak et al.(baseline)	100%	No	12.10	17.30
Beak et al.(w/o aug.; refine)	100%	No	10.40	16.40
Beak et al.(w/o refine)	100%	Yes, 10 times	9.10	14.90
Beak et al.	100%	Yes, 10 times	8.50	14.10
LSPS [124]	25%	No	7.35	15.70
	50%	No	7.10	15.45
	75%	No	7.05	15.45
	100%	No	7.00	15.40
Crossing Net [108]	25%	No	10.50	16.10
	50%	No	10.0	16.0
	75%	No	10.10	15.90
	100%	No	10.20	15.50
SO-HandNet [7]	25%	No	11.10	14.90
	50%	No	9.40	14.10
	75%	No	9.10	12.80
	100%	No	7.70	11.20
Ours	25%	No	**6.11**	**8.14**
	50%	No	**6.06**	**8.11**
	75%	No	**6.04**	**8.06**
	100%	No	**5.99**	**8.01**
Ours	1%	No	6.94	8.71

strates the practicality of the proposed method in cases of severely limited access to labeled data.

4.5.7 Comparison with State-of-the-Art Fully-Supervised Methods

We compare the proposed method with state-of-the-art methods that use 100% of ground-truth annotations for training [46, 45, 85, 91, 92, 94, 95, 11, 9, 8, 56, 49, 93, 51, 10, 73, 58, 52]. Table 4.9 summarizes the performance based on the mean distance error on the three datasets. As can be seen in Table 4.9, despite using only 25% of the ground-truth annotations, the proposed method ranks as the best performing method

Table 4.8: Comparison of our work with [4] under cases of severely limited access to labels on the NYU dataset. Numbers next to the methods represent their performance under the corresponding scenarios in terms of mean distance error in mm

Methods	Number of Labeled Examples			
	10	100	1,000	10,000
MURAUER [4]	**16.4**	12.2	10.90	9.90
Ours	25.82	**12.11**	**8.60**	**8.16**

Table 4.9: Comparison with the state-of-the-art fully-supervised methods on ICVL [1] (Left), NYU [2] (Middle), and MSRA [3] (Right). "Error" indicates the mean distance error in mm. 25p and 100p respectively denote the cases where 25% and 100% of the ground-truth annotations are used for training

Methods	Error
DeepModel [46]	11.56
DeepPrior [45]	10.40
DeepPrior++ [85]	8.10
REN-4x6x6 [91]	7.63
REN-9x6x6 [92]	7.31
DenseReg [11]	7.30
SHPR-Net [93]	7.22
HandPointNet [51]	6.94
CrossInfoNet [8]	6.73
NARHT [73]	6.47
A2J [9]	6.46
Point-to-Point [10]	6.30
V2V-PoseNet [52]	6.28
JGR-P2O [56]	6.02
HandFoldingNet [58]	5.95
Ours-25p	6.11
Ours-100p	5.99

Methods	Error
DeepPrior [45]	19.73
DeepModel [46]	17.04
3DCNN [49]	14.10
REN-4x6x6 [91]	13.39
REN-9x6x6 [92]	12.69
DeepPrior++ [85]	12.24
Pose-REN [94]	11.81
Generalized-Feedback [95]	10.89
HandPointNet [51]	10.54
DenseReg [11]	10.20
CrossInfoNet [8]	10.08
NARHT [73]	9.80
Point-to-Point [10]	9.10
A2J [9]	8.61
HandFoldingNet [58]	8.58
V2V-PoseNet [52]	8.42
JGR-P2O [56]	8.29
Ours-25p	8.14
Ours-100p	8.01

Methods	Error
REN-9x6x6 [92]	9.79
3DCNN [49]	9.58
DeepPrior++ [85]	9.5
Pose-REN [94]	8.65
HandPointNet [51]	8.5
CrossInfoNet [8]	7.86
SHPR-Net [93]	7.76
Point-to-Point [10]	7.7
V2V-PoseNet [52]	7.59
JGR-P2O [56]	7.55
NARHT [73]	7.55
HandFoldingNet [58]	7.34
DenseReg [11]	7.23
Ours-25p	7.28
Ours-100p	7.18

on the NYU dataset, the second best performing on the MSRA dataset, and the third best performing on the ICVL dataset. These observations clearly demonstrate the success of the proposed framework in significantly reducing the reliance on the labeled training data.

Algorithm 1 Semi-supervised hand pose estimation using the proposed method

Require: A network architecture f and two instantiations of its parameters θ_s and θ_T; hyper-parameters; the total number of epochs T_{max}; number of training iterations per epoch $R = N_s/B$, where N_s denotes the number of labeled examples

Initialize the thresholds T_j, the ratio ρ_t and λ_t
for t = 0, ... , T_{max} **do**
    for b = 0, ... , R **do**
        sample a batch of unlabeled examples x_u
        sample a batch of labeled examples $(x_l, \mathcal{J}^l)$
        $X_T = x_u \cup (x_l, \mathcal{J}^l)$
        $X_S = (\Phi(x_u), f(\Phi(x_u); \theta_{EMAN}))$
        Update θ_T via gradient descent of Eq. 4 from the original paper on X_T
        Update θ_S via gradient descent of Eq. 10 from the original paper on X_S
        Update θ_{EMAN}
    end for
    Update the thresholds T_j, the ratio ρ_t and λ_t
end for
Fine-tune θ_s using the available labeled examples
return Student network parameters θ_s

CHAPTER 5

Conclusions and Future Work

In this dissertation, we investigated two forms of hand analysis from depth images, namely 3D hand pose estimation and hand part segmentation. We first proposed a data-driven method that enables hand part segmentation without the need for the ground-truth segmentation maps for training. We then introduced a novel Neural Network architecture, that we called TriHorn-Net, to improve the accuracy of 3D hand pose estimation methods. Finally, in order to reduce heavy reliance of the existing 3D hand pose estimation methods on training data, we introduced a novel semi-supervised method based on a student-teacher framework to achieve a high accuracy using only a fraction of annotated data that the existing fully-supervised methods require for training.

We conclude this dissertation with a brief review of each chapter, summarizing the work and contributions therein. Discussions of future work are included as these challenges are ongoing and require further investigation before robust and accessible solutions can be provided.

5.1 Weakly-supervised Hand Part Segmentation

we presented the first data-driven method to perform hand part segmentation on depth images. We investigated the possibility of taking advantage of weak labels (in this case 3D joint locations) to learn the task of hand part segmentation. Thus, our method does not impose any additional burden in terms of requiring extra effort

to manually label data, which could be both expensive and labor intensive. Both quantitative and qualitative results demonstrate the effectiveness of our method.

The proposed method could have many potential applications that we have not investigated, including but not limited to shape estimation which is used in animation, gesture recognition and Augmented/Virtual reality. Our work opens new fronts for future research, including extending the proposed method to RGB images, which are more widely used in real-world scenarios. The proposed method also opens up some possibilities in terms of improving the performance of 3D hand pose estimation methods by incorporating part segmentation labels.

5.2 Fully-supervised 3D Hand Pose Estimation

We proposed TriHorn-Net, a novel and powerful neural network for 3D hand pose estimation from a single depth image. It achieves improved accuracy in hand pose estimation by introducing a novel formulation to decompose the 3D hand pose estimation into the estimation of 2D joint location in the image coordinate space, and the estimation of their corresponding depth values, which is guided by an attention map resulted from the fusion of two complementary attention maps computed by two separate branches. Experimental results on three challenging benchmarks demonstrate that, despite having a simple architecture and requiring no optimization approaches at test time, the proposed network outperforms the state-of-the-art methods. We also proposed a simple data augmentation method for depth-based hand pose estimation methods and presented empirical results demonstrating its effectiveness.

This work provides many opportunities for future research, including investigation of the ways the two complementary attention maps interact, which could lead to further improvements. Another line of research to build up on this work is in-

corporating more advanced means of computing attention maps (such as multi-head attention used in Transformers) into the proposed model.

5.3 Semi-supervised 3D Hand Pose Estimation

We proposed a novel framework for performing depth-based 3D hand pose estimation under scenarios where the access to the labeled data is limited. The proposed framework consists of two identical networks that are jointly trained. The teacher network is trained using consistency training on both labeled and unlabeled examples by adapting some of latest advancements in SSL methods in image classification. The student network is trained using the unlabeled examples and their corresponding pseudo-labels provided by the teacher network. After training, the teacher network is discarded and only the student network is used for inference. Extensive experiments demonstrate the proposed framework outperforms the current state-of-the-art methods by large margins under different scenarios in terms of the availability of the labeled data.

Since the heavy requirement for the labeled data for training remains to be a major bottleneck in the process of deploying 3D hand pose estimation methods based on Deep Learning, this line of research is expected to expand and be paid more attention in the future. One future research direction we suggest is to incorporate self-supervised representation learning methods into the proposed framework to further reduce the reliance on the labeled data.

REFERENCES

[1] D. Tang, H. Jin Chang, A. Tejani, and T.-K. Kim, "Latent regression forest: Structured estimation of 3d articulated hand posture," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3786–3793.

[2] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 5, pp. 1–10, 2014.

[3] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 824–832.

[4] G. Poier, M. Opitz, D. Schinagl, and H. Bischof, "Murauer: Mapping unlabeled real data for label austerity," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1393–1402.

[5] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Yong Chang, K. Mu Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, *et al.*, "Depth-based 3d hand pose estimation: From current achievements to future goals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2636–2645.

[6] J. Malik, I. Abdelaziz, A. Elhayek, S. Shimada, S. A. Ali, V. Golyanik, C. Theobalt, and D. Stricker, "Handvoxnet: Deep voxel-based network for 3d hand shape and pose estimation from a single depth map," in *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7113–7122.

[7] Y. Chen, Z. Tu, L. Ge, D. Zhang, R. Chen, and J. Yuan, "So-handnet: Self-organizing network for 3d hand pose estimation with semi-supervised learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6961–6970.

[8] K. Du, X. Lin, Y. Sun, and X. Ma, "Crossinfonet: Multi-task information sharing based hand pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9896–9905.

[9] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 793–802.

[10] L. Ge, Z. Ren, and J. Yuan, "Point-to-point regression pointnet for 3d hand pose estimation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 475–491.

[11] C. Wan, T. Probst, L. Van Gool, and A. Yao, "Dense 3d regression for hand pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5147–5156.

[12] S. Baek, K. In Kim, and T.-K. Kim, "Augmented skeleton space transfer for depth-based hand pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8330–8339.

[13] A. K. Bojja, F. Mueller, S. R. Malireddi, M. Oberweger, V. Lepetit, C. Theobalt, K. M. Yi, and A. Tagliasacchi, "Handseg: An automatically labeled dataset for hand segmentation from depth images," in *2019 16th Conference on Computer and Robot Vision (CRV)*. IEEE, 2019, pp. 151–158.

[14] A. Boukhayma, R. d. Bem, and P. H. Torr, "3d hand shape and pose from images in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 843–10 852.

[15] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 6, p. 245, 2017.

[16] A. Spurr, J. Song, S. Park, and O. Hilliges, "Cross-modal deep variational hand pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 89–98.

[17] Y. Cai, L. Ge, J. Cai, and J. Yuan, "Weakly-supervised 3d hand pose estimation from monocular rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 666–682.

[18] D. Kulon, R. A. Guler, I. Kokkinos, M. M. Bronstein, and S. Zafeiriou, "Weakly-supervised mesh-convolutional hand reconstruction in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4990–5000.

[19] C. Wan, T. Probst, L. V. Gool, and A. Yao, "Self-supervised 3d hand pose estimation through training by fitting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 853–10 862.

[20] A. Urooj and A. Borji, "Analysis of hand segmentation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4710–4719.

[21] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1949–1957.

[22] A. A. Argyros and M. I. Lourakis, "Real-time tracking of multiple skin-colored objects with a possibly moving camera," in *European Conference on Computer Vision*. Springer, 2004, pp. 368–379.

[23] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, 2002.

[24] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern recognition*, vol. 40, no. 3, pp. 1106–1122, 2007.

[25] S. Kim, H.-g. Chi, X. Hu, A. Vegesana, and K. Ramani, "First-person view hand segmentation of multi-modal hand activity video dataset."

[26] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1219–1225.

[27] A. Fathi, X. Ren, and J. M. Rehg, "Learning to recognize objects in egocentric activities," in *CVPR 2011*. IEEE, 2011, pp. 3281–3288.

[28] C. Li and K. M. Kitani, "Pixel-level hand detection in ego-centric videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3570–3577.

[29] E. Ardizzone, M. La Cascia, and D. Molinelli, "Motion and color-based video indexing and retrieval," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 3. IEEE, 1996, pp. 135–139.

[30] B. Kang, K.-H. Tan, N. Jiang, H.-S. Tai, D. Tretter, and T. Nguyen, "Hand segmentation for hand-object interaction from depth map," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 259–263.

[31] G. L. Oliveira, A. Valada, C. Bollen, W. Burgard, and T. Brox, "Deep learning for human part discovery in images," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1634–1641.

[32] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 3633–3642.

[33] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, *et al.*, "Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.

[34] D. Joseph Tan, T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, and J. Shotton, "Fits like a glove: Rapid and reliable hand shape personalization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5610–5619.

[35] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BmVC*, vol. 1, no. 2, 2011, p. 3.

[36] A. Tkach, M. Pauly, and A. Tagliasacchi, "Sphere-meshes for real-time hand modeling and tracking," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 6, pp. 1–11, 2016.

[37] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt, "Real-time hand tracking using a sum of anisotropic gaussians model," in *2014 2nd International Conference on 3D Vision*, vol. 1. IEEE, 2014, pp. 319–326.

[38] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon, "Learning an efficient model of hand shape variation from depth images," in

Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 2540–2548.

[39] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015.

[40] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3907–3916.

[41] L. Kavan and J. Žára, "Spherical blend skinning: a real-time deformation of articulated models," in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 2005, pp. 9–16.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[44] L. Duan, M. Shen, S. Cui, Z. Guo, and O. Deussen, "Estimating 2d multi-hand poses from single depth images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[45] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands deep in deep learning for hand pose estimation," *arXiv preprint arXiv:1502.06807*, 2015.

[46] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei, "Model-based deep hand pose estimation," *arXiv preprint arXiv:1606.06854*, 2016.

[47] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a feedback loop for hand pose estimation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3316–3324.

[48] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Heloir, and D. Stricker, "Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 110–119.

[49] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "3d convolutional neural networks for efficient and robust hand pose estimation from single depth images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1991–2000.

[50] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.

[51] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8417–8426.

[52] G. Moon, J. Y. Chang, and K. M. Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in *Proceedings of the IEEE conference on computer vision and pattern Recognition*, 2018, pp. 5079–5088.

[53] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Hand pose estimation and hand shape classification using multi-layered randomized decision forests," in *European Conference on Computer Vision*. Springer, 2012, pp. 852–863.

[54] S. Li and D. Lee, "Point-to-pose voting based hand pose estimation using residual permutation equivariant layer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 927–11 936.

[55] P. Ren, H. Sun, Q. Qi, J. Wang, and W. Huang, "Srn: Stacked regression network for real-time 3d hand pose estimation." in *BMVC*, 2019, p. 112.

[56] L. Fang, X. Liu, L. Liu, H. Xu, and W. Kang, "Jgr-p2o: joint graph reasoning based pixel-to-offset prediction network for 3d hand pose estimation from a single depth image," in *European Conference on Computer Vision*. Springer, 2020, pp. 120–137.

[57] U. Iqbal, P. Molchanov, T. B. J. Gall, and J. Kautz, "Hand pose estimation via latent 2.5 d heatmap regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 118–134.

[58] W. Cheng, J. H. Park, and J. H. Ko, "Handfoldingnet: A 3d hand pose estimation network using multiscale-feature guided folding of a 2d hand skeleton," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 260–11 269.

[59] V. Athitsos and S. Sclaroff, "Estimating 3d hand pose from a cluttered image," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2. IEEE, 2003, pp. II–432.

[60] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4903–4911.

[61] X. Tang, T. Wang, and C.-W. Fu, "Towards accurate alignment in real-time 3d hand-mesh reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 698–11 707.

[62] Y. Zhou, M. Habermann, W. Xu, I. Habibie, C. Theobalt, and F. Xu, "Monocular real-time hand shape and motion capture using multi-modal data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5346–5355.

[63] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan, "3d hand shape and pose estimation from a single rgb image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 10 833–10 842.

[64] S. Baek, K. I. Kim, and T.-K. Kim, "Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1067–1076.

[65] X. Zhang, H. Huang, J. Tan, H. Xu, C. Yang, G. Peng, L. Wang, and J. Liu, "Hand image understanding via deep multi-task learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 281–11 292.

[66] G. Moon, T. Shiratori, and K. M. Lee, "Deephandmesh: A weakly-supervised deep encoder-decoder framework for high-fidelity hand mesh modeling," in *European Conference on Computer Vision*. Springer, 2020, pp. 440–455.

[67] G. Moon and K. M. Lee, "I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*. Springer, 2020, pp. 752–768.

[68] A. Spurr, U. Iqbal, P. Molchanov, O. Hilliges, and J. Kautz, "Weakly supervised 3d hand pose estimation via biomechanical constraints," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*. Springer, 2020, pp. 211–228.

[69] A. Tkach, A. Tagliasacchi, E. Remelli, M. Pauly, and A. Fitzgibbon, "Online generative model personalization for hand tracking," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 6, pp. 1–11, 2017.

[70] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall, "Capturing hands in action using discriminative salient points and physics simulation," *International Journal of Computer Vision*, vol. 118, no. 2, pp. 172–193, 2016.

[71] H. Liang, J. Yuan, and D. Thalmann, "Parsing the hand in depth images," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1241–1253, 2014.

[72] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton, "Opening the black box: Hierarchical sampling optimization for estimating human hand pose," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3325–3333.

[73] L. Huang, J. Tan, J. Liu, and J. Yuan, "Hand-transformer: non-autoregressive structured modeling for 3d hand pose estimation," in *European Conference on Computer Vision*. Springer, 2020, pp. 17–33.

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[75] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[76] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," *arXiv preprint arXiv:1804.09541*, 2018.

[77] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[78] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[79] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[80] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[81] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.

[82] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[83] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*. Springer, 2016, pp. 483–499.

[84] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 529–545.

[85] M. Oberweger and V. Lepetit, "Deepprior++: Improving fast and accurate 3d hand pose estimation," in *Proceedings of the IEEE international conference on computer vision Workshops*, 2017, pp. 585–594.

[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[87] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[88] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[89] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3349–3364, 2020.

[90] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 466–481.

[91] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yang, "Region ensemble network: Improving convolutional network for hand pose estimation," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 4512–4516.

[92] G. Wang, X. Chen, H. Guo, and C. Zhang, "Region ensemble network: Towards good practices for deep 3d hand pose estimation," *Journal of Visual Communication and Image Representation*, vol. 55, pp. 404–414, 2018.

[93] X. Chen, G. Wang, C. Zhang, T.-K. Kim, and X. Ji, "Shpr-net: Deep semantic hand pose regression from point clouds," *IEEE Access*, vol. 6, pp. 43 425–43 439, 2018.

[94] X. Chen, G. Wang, H. Guo, and C. Zhang, "Pose guided structured region ensemble network for cascaded hand pose estimation," *Neurocomputing*, vol. 395, pp. 138–149, 2020.

[95] M. Oberweger, P. Wohlhart, and V. Lepetit, "Generalized feedback loop for joint hand-object pose estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 8, pp. 1898–1912, 2019.

[96] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[97] A. Demiriz, K. P. Bennett, and M. J. Embrechts, "Semi-supervised clustering using genetic algorithms," *Artificial neural networks in engineering (ANNIE-99)*, pp. 809–814, 1999.

[98] A. Gammerman, V. Vovk, and V. Vapnik, "Learning by transduction," *arXiv preprint arXiv:1301.7375*, 2013.

[99] T. Joachims *et al.*, "Transductive inference for text classification using support vector machines," in *Icml*, vol. 99, 1999, pp. 200–209.

[100] Y. Grandvalet, Y. Bengio, *et al.*, "Semi-supervised learning by entropy minimization." *CAP*, vol. 367, pp. 281–296, 2005.

[101] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.

[102] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the ninth international conference on Information and knowledge management*, 2000, pp. 86–93.

[103] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *International Conference on Computational Learning Theory*. Springer, 2004, pp. 624–638.

[104] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," 2001.

[105] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 784–800.

[106] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 55–67, 2007.

[107] J. Li, C. Xiong, and S. C. Hoi, "Comatch: Semi-supervised learning with contrastive graph regularization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9475–9484.

[108] C. Wan, T. Probst, L. Van Gool, and A. Yao, "Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 680–689.

[109] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," *arXiv preprint arXiv:1904.12848*, 2019.

[110] Z. Cai, A. Ravichandran, S. Maji, C. Fowlkes, Z. Tu, and S. Soatto, "Exponential moving average normalization for self-supervised and semi-supervised learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 194–203.

[111] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[112] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020.

[113] J. S. Supancic, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, "Depth-based hand pose estimation: data, methods, and challenges," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1868–1876.

[114] L. Yang, S. Chen, and A. Yao, "Semihand: Semi-supervised hand pose estimation with consistency," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 364–11 373.

[115] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[116] J. Han, P. Luo, and X. Wang, "Deep self-learning from noisy labels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5138–5147.

[117] P. Cascante-Bonilla, F. Tan, Y. Qi, and V. Ordonez, "Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning," *arXiv preprint arXiv:2001.06001*, 2020.

[118] C. Gong, D. Wang, and Q. Liu, "Alphamatch: Improving consistency for semi-supervised learning with alpha-divergence," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 683–13 692.

[119] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE*

transactions on pattern analysis and machine intelligence, vol. 41, no. 8, pp. 1979–1993, 2018.

[120] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *arXiv preprint arXiv:1703.01780*, 2017.

[121] B. Zhang, Y. Wang, W. Hou, H. Wu, J. Wang, M. Okumura, and T. Shinozaki, "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[122] I. Nassar, S. Herath, E. Abbasnejad, W. Buntine, and G. Haffari, "All labels are not created equal: Enhancing semi-supervision via label grouping and co-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7241–7250.

[123] J. He, A. Kortylewski, S. Yang, S. Liu, C. Yang, C. Wang, and A. Yuille, "Rethinking re-sampling in imbalanced semi-supervised learning," *arXiv preprint arXiv:2106.00209*, 2021.

[124] M. Abdi, E. Abbasnejad, C. P. Lim, and S. Nahavandi, "3d hand pose estimation using simulation and partial-supervision with a shared latent space," *arXiv preprint arXiv:1807.05380*, 2018.

BIOGRAPHICAL STATEMENT

Mohammad Rezaei received his Bachelor's degree in Software Engineering from Isfahan University of Technology in 2014, and his Master's degree in Artificial Intelligence from Khajeh Nasir Toosi University of Technology in 2017. In 2018, he started his studies to pursue his Ph.D in Computer Science at the University of Texas at Arlington. His current research interests are in the areas of Computer Vision, Deep Learning and Machine Learning.