Instant Hand Pose Estimation and Reflection on Simulated Robot hands

Hongyi Bian, Boyan Li, Yuxiang Zhou, Haoquan Zhou *

Abstract

Hand Pose Estimation has drawn people's attention in recent years. Significant progresses have been made in this field with various of solutions provided by researches. Meanwhile, the development of robotics require robots to perform human-like behaviors precisely. In this paper, we proposed Sakura Net, which is a Deep Convolutional Neural Network inspired by existing TriHorn-Net. Several prevalent components including three-branch bottleneck structure and Swin Transformer are used in our network in order to improve the performance. We show that our network can estimate hand poses in a precise manner from a depth image. We also demonstrate that our model can map the estimated hand pose to simulated robot hands with relatively high accuracy.

Keywords *Hand Pose Estimation, Depth Image, Transformer, Robotics, Robot hand imitating hand poses*

1. Introduction

Recent advances in motion capturing area make it possible to capture detailed human body actions using videos or pictures shot by cameras. As one core part of human's body language, the hand pose and its estimation have been drawing researchers' attention in recent years. Hand pose estimation requires the computer to find the joints or key points of the hand from an image or a set of videos, so that the position and orientation of the whole hand can be estimated. The captured hand pose can be further used to accomplish tasks like constructing live2d figures or set up a Human-Computer Interface (HCI) [1].

Meanwhile, the development of robotics requires increasingly precise movement of robots. Under some circumstances, robots are asked to imitate the exact behaviors of human in order to accomplish certain tasks like fetching an egg without slipping nor squeezing. Such tasks require a high accuracy when copying human's hand poses.

Thus, it is of great value to explore the use of in-time hand pose estimation techniques and map the captured poses precisely to the corresponding movement of robots. In this research project, we specifically focus on mapping the gestures of human hands described by depth pictures to the corresponding parts of robot hands. We will not consider situations when input pictures are in RGB scale. We are also not interested in the transition process from one hand pose to another hand pose.

In fact, there are various methods provided by researchers worldwide to solve the hand pose estimation problem. Before the introduction of deep learning techniques, the handcrafted features are applied to extract the hand poses. For example, Athitsos presented an image-to-model chamfer distance with general-purpose, probabilistic line matching method to capture the line segments of hands [2]. Another method called Particle Swhand Optimization (PSO) is efficient in extracting hand poses as well [3]. Traditional machine learning algorithms like Random Forest are also used to implement classification tasks for general hand pose estimation [4].

After the Convolutional Neural Networks (CNNs) proved its ability in deep learning area, researchers turned to this model and developed new Deep Neural Networks (DNNs) models like Region Ensemble Network (REN) [5], Hand pointnet [6], and Anchor-to-joint Regression Network (A2J) [7]. These deep learning models significantly improve the performance of hand pose estimation.

The task of hand pose estimation based on depth image can be defined as follows: given an input depth image $P_I \in \mathbb{R}^{m \times n}$, where m, n denotes the height and width of the picture respectively, find the 3D locations of k pre-defined key points or joints $J_k = (U, V, D) \in \mathbb{R}^3$. That is, the output will be a matrix $P_O \in \mathbb{R}^{k \times 3}$.

To solve this problem, we propose Sakura Net, which is a deep convolutional neural network inspired by the existing TriHorn-Net [8]. We constructed our network referring to the overall idea of TriHorn-Net and show that our model outperforms the TriHorn-Net by 5%. In addition, We map the estimated hand poses to a simulated robot hand generated by Solidworks & Matlab.

2. Related Work

2.1. Hand Pose Estimation

There are several representative works in hand pose detection. One of the most classic solution of hand pose esti-

*This submission is the project progress report for course ECE4880J in 2022SU. This course is held by UM-SJTU Joint Institute, Shanghai Jiao Tong University. The mentor of this course is Dr. Siheng Chen.

mation is the Region Ensemble Network (REN) proposed by Guo et al [5]. They demonstrate a CNN-based architecture with region ensemble strategy. They divide the input hand picture into different regions and feed each region into the corresponding CNN respectively. The outputs of the CNNs are put together to form the final result. This model is proved to outperform most of the traditional CNN-based methods at that time. However, by dividing the input picture into different parts, the model will lose some global information which may contribute to the final result.

The Anchor-to-Joint Regression Network reaches a relatively fast computation speed while maintaining high performance [7]. The traditional works are usually based on 3D CNNs to detect the hand pose. A2J makes full use of the depth information in 2D images and implements an end-to-end 2D CNNs-based deep learning model. The A2J model is relatively easy to train and thus can be trained using large-scale data set. But the flaw for this model is that the backbone of the model is purely based on ResNet. The additional work is mainly focus on the post processing part and tried to predict three coordinates (U, V, D), of joints at the same time. This structure does not pay enough attention to the relations between the position coordinates (D, V) and the depth coordinate D.

The disentangled variational autoencoder (dVAE) is used to estimate the hand poses from RGB images [9]. The dVAE can synthesize highly realistic images of the hand specifiable by both pose and image background content and also estimate 3D hand poses from RGB images.

A method called HandAugment to synthesize image data to augment the training process of the neural networks is also proposed by Zhang [10]. This method set one neutral network to make the training process focus on the hand regions and thus to improve the performance. The problem is that this model is only tested to be the state-of-the art solution on the data set Hand 2019.

Other methods focused on the hand pose detection with depth images. Rezaei et al. [8] decompose depth image into depth image space (UV), attention enhance branch and depth branch. The former two branches focus on the positions of the joints using attention maps, while the third branch focus on the depth information. But these deep neural network based methods may fail with severe occlusion occurs. Also, the fully connected layer used to output the predicted coordinates has the same parameters for all joints, which may not be precise enough.

Huang et al. [11] try to solve this problem by combining the deep neural network based methods with the detection based methods. An adaptive weighting regression is proposed to make the detection based parts learnable and the two parts are trained separately to give more accurate and robust results. This model also considers the output coordinates (U, V, D) at the same time, and may neglect the relations between position coordinates (U, V) and the depth coordinate D.

2.2. Robot hand Imitating Hand Poses

The traditional way to deal with this problem can contain the methods like inverse kinematics and optimizing methods. However the work space of robotic hand is much smaller than the human hand, so the inverse kinematics methods cannot ensure a proper result. The optimization methods can always give quasi-optimal results. Handa et. al [12] use SLSQP to optimize the cost function, but the computational cost can be unaffordable.

As a matter of fact, neural network methods are mentioned. Li tried to use encoder-decoder method to map the human hand motion to robotic arms, however, its training needs continuous figures of human as well as the corresponding robotic arm motions. However, there's actually very limited work that do this work using neural networks [13].

3. Approach

As the name of our model indicated, our Sakura Net is inspired by the TriHorn-Net proposed by [8]. The general model structure is shown in Fig. 1. The model is composed of three parts: the encoder, the bi-branches structure extracting position (U, V) information and depth D information, and the pooling & fully connected layer which predicts the coordinates of each joint. In the first part, the input depth image is fed into a Stacked Hourglass Model to extract features both in high-level and low-level. The pipeline is then branched into two routes. The UV branch focus on the position information of each joints, while the depth branch focus on the depth feature map of each joint. After that, feature pooling & fully connected layer will together merge the position and depth information to produce the final 3D coordinates (U, V, D) of each joint. Afterward, the output of joint positions will be treated as the input in the following imitation learning network and the give the optimized robotic arm pose.

3.1. Encoder

Encoder is a non-linear mapping from the input depth image to output features both in high-level and low level. In our model, the input is a depth image $P_I \in \mathbb{R}^{m \times n}$ and the output is a feature matrix $f \in \mathbb{R}^{256 \times w \times h}$, where (w, h) are the width and height of the output feature matrix. For each of the entry, we have 256 features. The detailed structure of the encoder is shown in Figure.2. The encoder used in our pipeline is a Stacked Hourglass Model proposed by [14]. Hourglass Network is a convolutional neural network using residual module as unit component. Each residual module contains three layers: a convolution layer, a batch normalization layer, and a ReLU layer. Each Hourglass Network

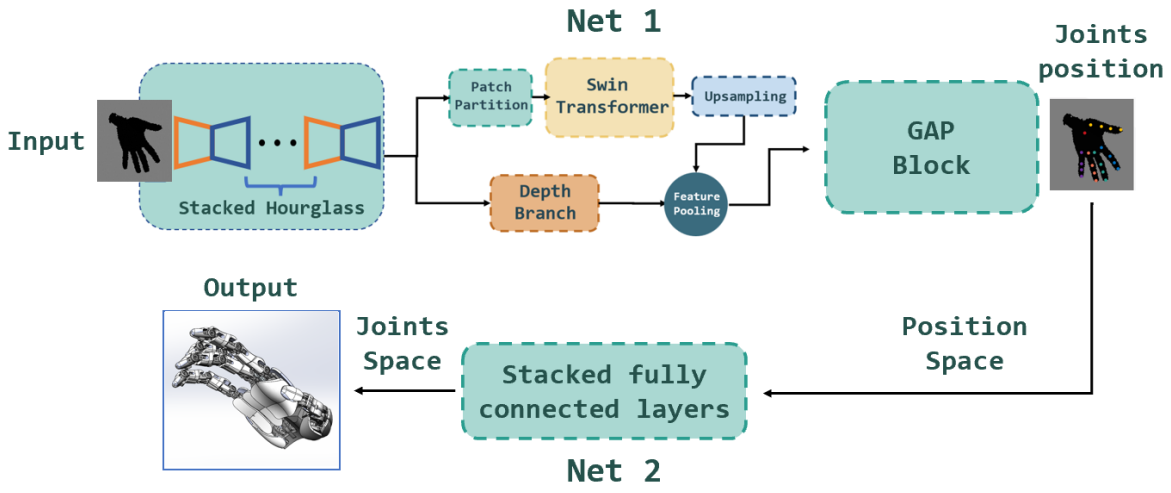

Figure 1. Overall Structure of Sakura Net

is then composed of one max pooling layer, three consecutive residual modules and an upsampling layer. Finally, by stacking Hourglasses together, a Stacked Hourglass Model can be constructed. In our model, different number of stacks is tested and the one with best performance is eventually selected.

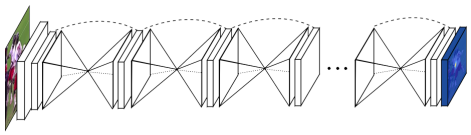

Figure 2. Stacked Hourglass Model [14]

3.2. 3-branch bottleneck structure

The building block of our network is delighted by the *Bottleneck* block which is used to construct ResNet-50 [15]. The mapping relationship in Bottleneck block is shown in equation 1, where $\mathcal{F}$ is a sequence of convolution-BatchNorm-ReLU layers w.r.t. the weight $\mathbf{W}$, and $\mathcal{I}$ is the identity map.

$$\mathbf{x_{out}} = \mathcal{I}(\mathbf{x_{in}}) + \mathcal{F}(\mathbf{x_{in}}, \mathbf{W}) \quad (1)$$

Instead of the two-branch structure in ResNet-50, inspired by the Hourglass Residual Unit proposed in [16], we implemented a 3-branch bottleneck block as the building block of our network. It is descibed in Equation 2.

$$\mathbf{x_{out}} = \mathcal{I}(\mathbf{x_{in}}) + \mathcal{F}(\mathbf{x_{in}}, \mathbf{W_f}) + \mathcal{P}(\mathbf{x_{in}}, \mathbf{W_p}) \quad (2)$$

In the 3-branch bottleneck block, one branch called *Pooling branch* $\mathcal{P}$ is added in comparison with the original bottleneck in ResNet-50. Pooling branch is composed of a maxpooling layer, followed by two convolution layers. After that, we use an upsampling layer to align the size.

Due to the sampling identity of the maxpooling layer, the pooling branch provides a channel to forward an aggregation of the features of the input from one range, which is defined by the kernel size of maxpooling layer. In this way, the reception field of the network structure in pooling branch will be enlarged. Therefore, the 3-branch bottleneck block will obtain aggregated high-level information while the detailed features are still remained by the branch $\mathcal{F}$ and $\mathcal{I}$.

The structure of 3-branch bottleneck structure is illustrated in Figure 3. The *ConvBlock* in the figure is the common unit used for convolution networks, composed of one convolution layer followed by batchNorm and ReLu, shown in Figure 4.

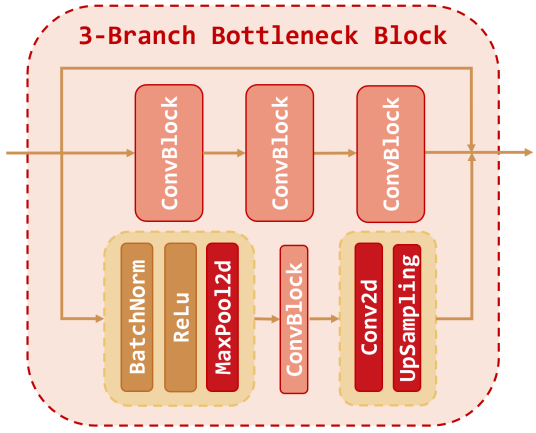

Figure 3. 3-branch Bottleneck Block

3.3. UV Branch

In our method, the features extracted by the Hourglass network are managed in two branches – UV branch and

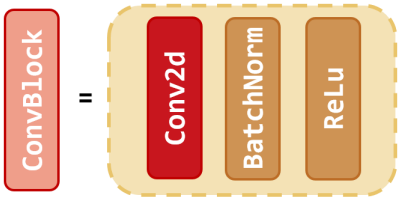

Figure 4. Convolution Block

depth branch. TriHorn-Net utilized two branches, which are UV branch and attention enhancement branch, to get the attention map separately and mix them together with learnable parameter. The reason is that, convolutional network work tend to focus on local information, so an extra branch is needed to grasp the global information. To tackle this problem, swin transformer [17] is utilized instead of the bottleneck structure in our UV branch. Swin transformer is adapted from Vision Transformer (ViT), which utilized the transformers technique in natural language processing (NLP) to the computer vision tasks. It takes the advantage of grasping global information. In our method, we first try the swin transformer, and the structure is shown in Fig. 5. Demonstrated in the figure, two transformer blocks will oper-

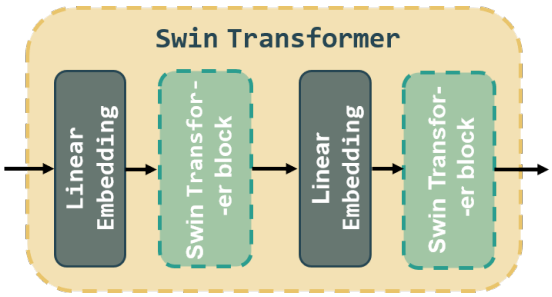

Figure 5. Swin Transformer

ate in series with linear embedding in between. The detailed swin transformer block structure is shown in Fig. 6

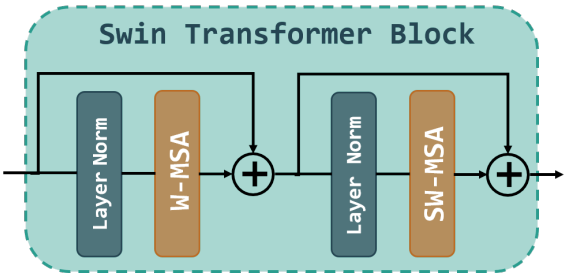

Figure 6. Swin Transformer Block

As shown in figure, with input $H \times W \times C$, swin transformer is first partitioned with patch size of 4×4. Here we use four successive Swin Transformer Blocks and got the output of $\frac{H}{4} \times \frac{W}{4} \times 4C$. Each Swin Transformer block consists of a shifted window based multi-head self attention (MSA) module, followed by a 2-layer Multilayer Perceptron (MLP) with GELU activation function in between. A LayerNorm (LN) layer is applied before the MSA module and MLP. A residual connection is applied after each MSA module and MLP. The output of swin transformer block is then recovered to the original image size by upsampling and projected to 16 channels heat maps, where each map represents the attention map for each joint. The UV position of the joints are calculated according to the equation:

$$(U^j, V^j) = \sum_{ui} \sum_{vi} (u_i, v_i) H_j^{2D}(u_i, v_i) \tag{3}$$

where H_j^{2D} is the the heat map normalized with softmax layer:

$$H_j^{2D} = \frac{exp(Att_{uv}^j(x,y)}{\sum_{u_i,v_i \in \Omega} exp(Att_{uv}^j(u_i,v_i))} \tag{4}$$

3.4. Depth Value Estimation

The depth value of the joints can be estimated from the feature vectors which are gained by the pooling process as follows:

$$F_j = Att_{\text{fused}}^j \circ \mathfrak{D} = \sum_x \sum_y Att_{fused}^j(x,y)\mathfrak{D}(x,y) \tag{5}$$

where j is the number of joints. Inspired by the Global Average Pooling (GAP) [18], we replace the fully connected layer by the GAP layer which takes the mean value of the feature vectors as the output illustrated in Fig.7 where z_j is the depth value of j^{th} joint. There are two advantaged of the GAP.First, the GAP is more simple and natural in the transformation between the feature map and the final depth value. Second, unlike the FC layer, there is no parameters to be trained in the GAP layer, which will make the model more robust and better anti-over-fitting effect.

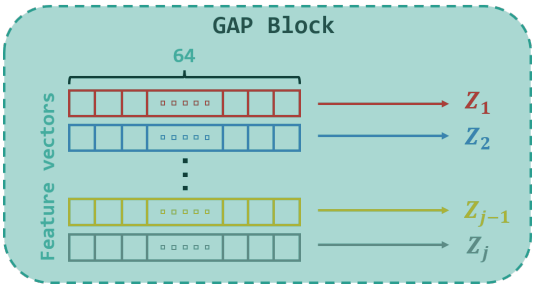

Figure 7. Global average pooling.

3.5. Robotic Hand Imitation Learning

In the robotic hand imitation learning part, we try to map the hand pose to the robotic hand which we obtained online [19] with neural networks. In this part, we take the output, which is the position of each joints, as an input. Then a multi-layer perception is applied to the input and the output is the pose parameter of the whole hand and the rotational angle of each joint. Afterward, the forward kinematics is applied to map the pose and angle output to the robotic hand's joint positions. One thing needs mentioning is that is the forward kinematics process, we use the dh-parameter to obtain the pose matrix and the position of each joint. And the detailed process is shown as follows. Firstly, due to the mechanical structure of the robotic hand, we treat the MCP points and the wrist point as a rigid body (refer to fig. 8). So the transformation matrix of these joints from the base coordinate can be calculated with Euler angle:

$$T_1^0 = \begin{bmatrix} c1+c3+s1s2s3 & c3s1s2-c1s3 & c2s1 & t_1 \\ c2s3 & c2c3 & -s2 & t_2 \\ c1s2s3-s1c3 & s1s3+c1c3s2 & c1c2 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where c represents cos, s represent sin. 1, 2, 3 means the roll, pitch, yaw angle and t_1, t_2, t_3 means the translation motion on x, y, z axis. Afterward, the transformation matrix between two adjacent joints can be obtained with dh-parameters, which is a standard method in robotics fields:

$$T_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & \alpha_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Finally, the pose matrix T and the joint position J for each joint can be calculated as:

$$\begin{aligned} T_i^0 &= T_1^0 T_2^1 ... T_i^{i-1} \\ J_i &= T_i^0 [[0,0,0,1]]^T \end{aligned} \tag{8}$$

In this way, we map the angle space to position space. The loss function is the L1 loss between the joint position for robotic hand and human hand. Finally, a network map the human hand joints to robotic hands is established.

4. Experiments

4.1. Datasets used

4.1.1 ICVL Dataset

The ICVL data set [20] contains 22K 320×280 depth images for training and 1.6K 320×280 depth images for testing. The label for each picture contains 16 joints. One joint is centered at the palm while the other 15 joints are equally distributed on five fingers. Each finger has three joints.

4.1.2 MSRA Dataset

MSRA data set [21] contains in total 9 subjects' right hands captured using Intel's Creative Interactive Gesture Camera. Each subject has 17 gestures captured and there are about 500 frames for each gesture with depth image's size of 320x240. The label for each picture has 21 3D points in (x, y, z) coordinates. The details can refer to 8 [13].

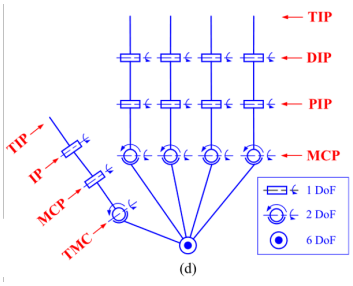

Figure 8. Joint points for MSRA Dataset

4.2. Evaluation method

We will use the mean distance error (MDE) measured in millimeter as the evaluation metric. The MDE is calcuated by

$$\begin{aligned} loss &= \sum_{J_i \in \text{test set}} ||\hat{J}_i - J_i||_2 \\ &= \sum_i \sqrt{(\hat{u}_i - u_i)^2 + (\hat{v}_i - v_i)^2 + (\hat{d}_i - d_i)^2} \end{aligned} \tag{9}$$

For robotic hand imitation learning part, we use MSRA as our input. A L1 loss between the joints position of human hands and robotic hands is used as the evaluation metric.

4.3. Baseline

For the hand-pose estimation part, we choose the TriHorn-Net trained using 20K pictures with 40 Epochs and 0.001 learning rate [8] as the baseline. Note that the number of Stacked Hourglasses in TriHorn-Net is by default set to 1. The mean distance error (in mm) for the baseline is 6.373.

For robotic hand imitation learning part, since there are very little research focusing on this field, and the majority of these researches takes the accuracy and the consuming time as the metric, we cannot give a very accurate baseline. One baseline that we can refer to is the work done by [22]. However, since we use a different dataset, we cannot directly use their results.

4.4. Experiment details

We conduct ablation experiment on each of the adjustments to the baseline model. We train the model of learning rate 0.001 and weight decay 2.0^{-5} on the training set of 20K pictures on one NVIDIA HGX A100 GPU. Moreover, the number of training epoches and training time are listed in Table 1. We also do an overall experiment together with all adjustments.

4.5. Results

4.5.1 Experiment of Number of Stacked Hourglasses

The test result for different number of Stacked Hourglasses on the test set is shown in Fig 9, while the minimum MDE

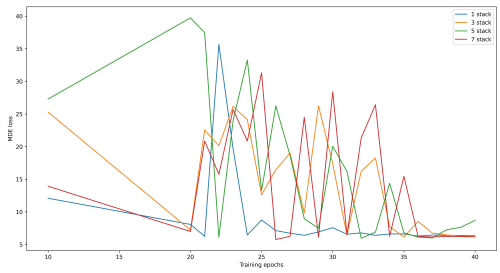

Figure 9. MDE loss with different stack number

loss for each stack number is shown in Table. 2

We find that as number of stacks increases for the Stacked Hourglass Network, the minimum MDE loss decreases. And all the MDE losses are smaller than the baseline after adding the number of stacks. However, the Fig 9 shows that the loss calculated on the test set is still quite unstable. Although there are improvements, we tend to believe that they are due to random chance.

4.5.2 Experiment of 3-bottle Bottleneck Block

In the implementation of 3-bottle bottleneck block, with only this basic unit altered, we observed the loss of the best model is 6.160, reached at 21st epoch. The converged loss is around 6.4 over the total training of 40 epochs. Compared with the baseline case 6.262, the improvement is not remarkable. This may mean that the original bottleneck is good enough to capture all the necessary features.

4.5.3 Experiment of Swin Transformer

In the experiment, we apply one and two swin transformer block. With other parameters unchanged, the new model give min loss of 6.771 for one block and 5.913 for two blocks. Higher number of swin transformer blocks is not applicable due the the dimensions of extracted features. Compared with the baseline case 6.262, our new model achieves a progress of 5.6%. This result shows that our new method can deal with the extracted features better.

the improvement is not remarkable. This may mean that the original bottleneck is good enough to capture all the necessary features.

4.5.4 Experiment of Global Average Pooling

For this experiment, only the fully connected layer is replaced by the global average pooling method and the parameters related to the fully connected layer are eliminated. The model with 40 epoch give us the best loss of 5.948 which may due to the randomness and the steady value is around 6.1.

4.5.5 Experiment of Robotic Hand Imitation Learning

For this part, actually we have worked out nearly all the code, but due to the time limitation and also the crowded server, we failed to return an output of our self-designed loss function with gradient. So we cannot train this model due to this problem. But since the bug only appears when the backward function is applied, which is not the main stream of our network, we believe that we can get good results if time permitted.

4.5.6 Overall Results

The overall result that applies all the features above is shown in Table 3. In the table, the all* represents for the combination for all features except 3-branch bottleneck block. We can see that, after combining all our features, the Sakura Net achieves a considerably good performance compared with other state-of-the-art methods. Also, it is worth mentioning that the overall model excluding 3-branch bottleneck block that contains all the features reaches its best performance at last training epoch.

5. Conclusion

5.1. Achievements and Contributions

We proposed a deep convolutional neural network called Sakura Net, which can perform precise hand pose estimation and projection to robot hands. To capture the joint information precisely, we apply a Swin Transformer in parallel with a depth information branch. Moreover, we add one pooling-sampling branch to the bottleneck unit to enlarge the receptive field. We also apply the Global Average Pooling to prevent the vanishing gradient problem, and increase the number of stacked Hourglass to boost the representativity. The estimated hand pose in position space is further passed into stacked fully connected layers to obtain a robot hand in

Table 1. Detailed information of the sub-experiments

	Number of epoches	Number of stacked Hourglasses
Number of Stacked Hourglasses	40	53.2min
3-branch Bottleneck Pooling	60	72.1min
Swin Transformer	40	44.0min
Global Average Pooling	60	53.4min

Table 2. Minimum MDE loss for different number of stacks

Number of stack	1	3	5	7
Minimum MDE loss	6.262	6.162	5.952	5.762
Corresponding epoch	21	35	32	26

Table 3. Overall results based on the ICVL data set

Name of network	Average 3D error
TriHorn-Net	**5.73**
HandFoldingNet	5.95
AWR	5.98
PixelwiseRegression	6.152
V2V-PoseNet	6.28
A2J	6.461
Pose-REN	6.8
TriHorn-Net (20k training set)	6.262
Our model (swin only)	5.913
Our model (GAP only)	5.948
Our model (all*, 30 epoches)	5.940
Our model (all*, 60 epoches)	5.855
Our model (all)	6.371

joints space. Experiments show that our work reaches relatively good performance in ICVL dataset. Our network is also a light-weight network providing potential solutions for mapping hand poses to robot hands and imitation learning.

5.2. Limitations

Our study has several limitations. For example, our model can only deal with depth image. However, most of the daily cameras capture RGB images. Also, it is relatively expensive to purchase depth image cameras. Additionally, since the work space of our robot hand is much smaller than human hand, some of the hand poses cannot be projected to the robot hand. In this conditions, we will make the pose on the robot as similar as it could using the imitation learning technique [23]. Due to the lack of computing resources, our network is rather small, which contains 14M trainable parameters. In contrast, prevalent networks contain more than 50M parameters.

5.3. Future Improvement

5.3.1 EfficientNet

In the experiment of number of Stacked Hourglasses, we find that blindly increasing the number of hourglasses will not significantly improve the network's performance. It is also worth mentioning that stacking too many hourglasses will make the encoder network too deep. To solve the this problem and seek for potential improvement, we notice that [24] provides a thorough rethink about the structure of ResNet and proposes EfficientNet, which takes the relation between the width and depth of the network into consideration. The author proves that by properly setting the ratio of the network width and network depth, the model requires less training parameters while yielding a even better performance. Thus, it is potentially a good idea to replace the hourglass encoder in our network with an EfficientNet.

5.3.2 Ensemble

Another potential improvement that can be adopted by our model is introducing the learning paradigm of neural network ensemble. This paradigm is first introduced and analyzed in [25]. Generally speaking, ensemble is a training method which requires several neural networks to be trained in parallel and independently. The final regression or classification result will be generated in a voting manner using weighted averaging. The author argues that the ensemble strategy will improve the result obtained from a single neural network with large possibility. In the future, the ensemble strategy may be applied in several ways to our model. One choice is performing ensemble through different regions of the image as proposed in [5]. Another potential choice is performing ensemble through different number of hourglasses.

5.3.3 Multi-Semantics Attention in Stacked Hourglass

In our implementation, the stacked Hourglass structure is simply increasing the network depth to improve representativity. A more advanced harnessing of these stacked structure is to embed the multi-semantics attention into each layer of hourglass, which is introduced in [16]. Imitating how human understand the picture, more attention on local information should be assigned to lower stacks; and higher stacks should obtain more attention on holistic features.

5.4. Acknowledgement

Thanks ECE4880J teaching group for their generous help!

References

[1] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 52–73, 2007. 1

[2] 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. 1

[3] 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. 1

[4] 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. 1

[5] 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. 1, 2, 7

[6] 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. 1

[7] 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/CVF International Conference on Computer Vision*, 2019, pp. 793–802. 1, 2

[8] M. Rezaei, R. Rastgoo, and V. Athitsos, "Trihorn-net: A model for accurate depth-based 3d hand pose estimation," *arXiv preprint arXiv:2206.07117*, 2022. 1, 2, 5

[9] L. Yang and A. Yao, "Disentangling latent hands for image synthesis and pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9877–9886. 2

[10] Z. Zhang, S. Xie, M. Chen, and H. Zhu, "Handaugment: A simple data augmentation method for depth-based 3d hand pose estimation," *arXiv preprint arXiv:2001.00702*, 2020. 2

[11] W. Huang, P. Ren, J. Wang, Q. Qi, and H. Sun, "Awr: Adaptive weighting regression for 3d hand pose estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 061–11 068. 2

[12] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox, "Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9164–9170. 2

[13] R. Li, H. Wang, and Z. Liu, "Survey on mapping human hand motion to robotic hands for teleoperation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 5, pp. 2647–2665, 2021. 2, 5

[14] 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. 2, 3

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 3

[16] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, "Multi-context attention for human pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1831–1840. 3, 7

[17] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022. 4

[18] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013. [Online]. Available: https://arxiv.org/abs/1312.4400 4

[19] R. wrist, "Robotic wrist," 2021, accessed 5 April 2022. https://grabcad.com/library/robotic-wrist-3. 5

[20] D. Tang, H. J. Chang, A. Tejani, and T. K. Kim, "Latent regression forest: Structured estimation of 3d articulated hand posture," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 5

[21] 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. 5

[22] S. Li, X. Ma, H. Liang, M. Görner, P. Ruppel, B. Fang, F. Sun, and J. Zhang, "Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 416–422. 5

[23] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016. 7

[24] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1905.11946 7

[25] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999. 7