Array 16 (2022) 100251

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

An improved hand gesture recognition system using keypoints and hand bounding boxes

Tuan Linh Dang [a,*], Sy Dat Tran [a], Thuy Hang Nguyen [a], Suntae Kim [b], Nicolas Monet [b]

[a] School of Information and Communications Technology, Hanoi University of Science and Technology, 1 Dai Co Viet road, Hai Ba Trung district, Hanoi, Vietnam
[b] Avatar, NAVER CLOVA, Buljeong-ro, Bundang-gu, Seongnam-si, South Korea

ARTICLE INFO

Keywords:
HRNet
Hand gesture recognition
CNN
Two-pipeline architecture
Keypoints

ABSTRACT

Hand gesture recognition is a significant problem for human–computer interaction. One form of hand gesture recognition is static hand gestures. This study developed a static hand gesture recognition system, consisting of three modules: Feature extraction Module, Processing Module, and Classification Module. The feature extraction module uses human pose estimation with a top-down method to extract not only the keypoints but also body and hand bounding boxes. After being normalized and processed in the processing module, its output will be used as the input for the classification module in which we proposed an architecture called Two-pipeline architecture. In this module, we also employ different methods to find the most suitable one for this task. Experiments were conducted on three datasets called HANDS, OUHANDS, and SHAPE. Results showed that the proposed Two-pipeline architecture with 2.5 million parameters obtained accuracy of 94%, 98%, and 94% on three datasets. In addition, the lightweight version with 0.22 million parameters also achieved accuracy of 91%, 94%, and 96%.

1. Introduction

Gesture recognition is a central field in computer vision that tries to develop the interactions between machines and humans. It is non-verbal communication. Gestures include face movement, hand movement, and body movement.

Hand gesture recognition(HGR) is a subfield of human gesture recognition that focuses on the interaction of the human hand with the machine. HGR systems aim at convenience, speed, and cost-saving for users. With HGR, humans can effortlessly operate complex machines using only hands or fingers without physical contact at a long distance. In addition, there are many other applications of HGR such as sign languages [1], robot control [2], and gaming interface [3].

There are two forms of HGR, including dynamic hand gestures and static hand gestures. The dynamic HGR aims to recognize actions through videos, and the static HGR tries to recognize gestures in images. Not all gesture recognition problems need to be solved with dynamic gestures because some gestures can be recognized using static images. In addition, the dynamic HGR may not be implemented in resources-constrained devices such as mobile devices. That is why most effects for camera applications use static gestures. For example, static gestures help people control the camera to take a photo or help to play the next song while listening to music at a distance compared to the phone without any physical touch on the screen. Besides, static HGR can be deployed to edge devices without the need for strong hardware like GPU or CPU. Another example showing the usefulness of static HGR is a human–computer interface that is portable and lightweight to help people with hearing disabilities. Therefore, this study focuses on solving the static HGR task.

For the last few years, pose estimation has been developed, which is a task of finding joints of hand [4], face [5], head [6], and whole body [7]. These joints may recreate human gestures or actions with enough important information. Thus, hand gestures can be recognized using the joints. In addition, some studies focus on the more lightweight and higher-accuracy version of the pose estimation. Therefore, using pose estimation to process data in gesture recognition tasks has attracted many researchers.

Our proposed system uses pose estimation for processing the data. The data after processing, keypoints, is used as a type of input of different architectures for gesture recognition. This approach may solve the limitations of the previous studies. Pose estimation can give good results in complex backgrounds and bad conditions such as different light conditions and distances between the camera and hands. Therefore, the results of recognition can be improved. On the other hand, users do not need to wear any sensor or device on their bodies. This manuscript also proposes a two-pipeline architecture that can combine

* Corresponding author.
E-mail address: linhdt@soict.hust.edu.vn (T.L. Dang).

https://doi.org/10.1016/j.array.2022.100251
Received 14 February 2022; Received in revised form 30 June 2022; Accepted 21 September 2022
Available online 29 September 2022

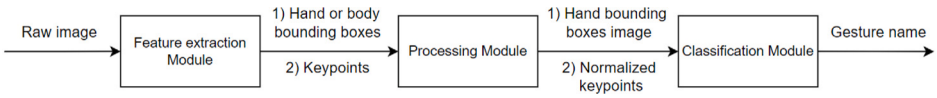


Fig. 1. System overview.

the features of both keypoints and images of the hand and body bounding boxes.

The experiments were conducted on three datasets called HANDS [8], OUHANDS [9], and SHAPE [10] datasets. These datasets contain hands that are both far and close to the camera, with various backgrounds and lighting conditions to investigate different situations.

Our main contribution to this manuscript can be listed as follows.

- Propose a two-pipeline architecture that can learn the feature combined from images and keypoints for prediction.
- Experiments on some datasets to find the best approach to the HGR problem.

This paper is presented as follows. The related work is shown in Section 2. Section 3 details our proposed architecture. The experimental results are demonstrated in Section 4. Finally, Section 5 concludes this paper.

2. Related work

2.1. Hand gesture recognition

A previous study has proposed a real-time hand gesture recognition system using a skin detector [11]. However, this system may experience difficulties in identifying human hands in difficult conditions, such as when the background color looks similar to the skin color. Another research has also introduced an HGR system using multiscale color features. Experimental results showed that the system performance was improved by conducting feature detection in color space. The system was implemented in a real-time approach [12]. However, the limit is that the system can only be used in real-time if there is no appearance of any object with other skin colors in the frame. Besides, segmentation has also been used in a two-stage convolution neural network (CNN) architecture to improve the recognition performance by learning features from both the RGB image and segmentation map before classification in HGR [13]. In this architecture, the first stage identifies the region of images that contains the hand, and the second stage classifies the gestures. Although this method has improved the recognition results compared to conventional segmentation methods, the accuracy was still not high. Other authors have used view projection from point cloud for the HGR [14]. This research first captures the points cloud using a depth sensor. After that, a CNN architecture is used to extract features. The final step is a support vector machine utilized as a classifier for gesture recognition. Recently, with the development of smart wearable devices, another approach for this problem is to use sensors to capture the hand coordinates and then use a machine-learning model to classify hand gestures based on machine-signal transformed from those coordinates [15]. However, this could not be a convenient approach because many users do not want to wear many sensors or wearable devices on their bodies.

2.2. Background

2.2.1. HRNet

The applications of pose estimation can be seen in tracking human pose [16], human sign language recognition [1], or human actions recognition [17]. Recently, HRNet [18] is one of the networks that achieved high accuracies in pose estimation tasks for detecting human keypoints. There are three different versions of HRNet called HRNetV1, HRNetV2, and HRNetV1p [19] which were respectively applied for human pose estimation, semantic segmentation, and object detection. The special feature of HRNet is that it can maintain high-resolution representations in the processing phase. The results are more spatially accurate by connecting the high-to-low resolution convolution streams and repeatedly exchanging information across resolutions. The pose estimation results on the COCO dataset show that HRNet reached significantly better results than the others [18].

Besides, object detection is one of the pose estimation steps before applying HRNet to extract keypoints. The authors of HRNet used MMDetection [20] in this step. This tool supports popular and contemporary detection frameworks such as Mask R-CNN, Faster R-CNN, and RetinaNet.

2.2.2. MobileNetV2

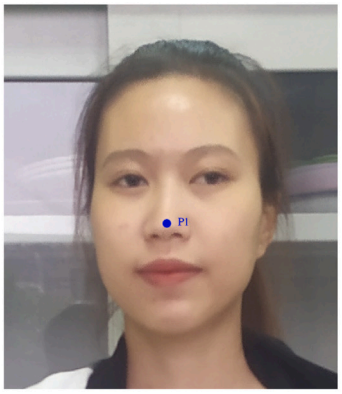
MobileNetV2 is a lightweight neural network architecture designed for mobile and resource-restricted environments. This architecture is based on an inverted residual structure. In addition, there are depthwise convolutions in the middle expansion layer. The inverted residual with a linear bottleneck is a notable layer of MobileNetV2, considered one of the most popular architectures for developing machine learning applications on resource-constrained devices with reduced memory cost and inference time while still preserving good performance. MobileNetV2 has been used to solve problems such as Face attribute detection [21], Hand Gesture Recognition [22], or Mask Detection [23].

2.2.3. Fully-connected neural network (FCN)

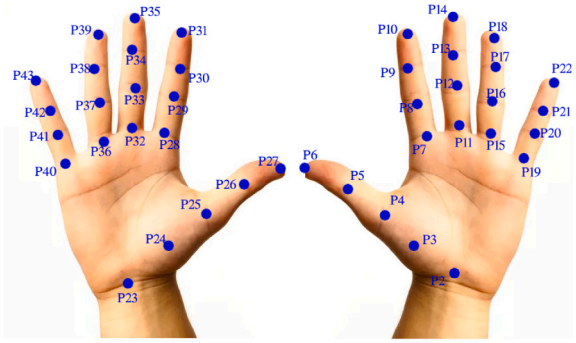
An FCN consists of several consecutive fully-connected layers, where all nodes from this layer are connected to all nodes in the next layers. Normally, an FCN is used to learn non-linear high-level features that are the outputs of the convolution layers. Similar to CNN, the FCN has an input layer, hidden layers, and an output layer. Some authors even demonstrated that the FCN could be converted to have the same calculation as a convolutional neural network using matrix multiplication [24].

3. Proposed method

An overview of our system is shown in Fig. 1 with three main modules including the feature extraction module, the processing module, and the classification module. The feature extraction module uses MMDetection [20] to detect hand or body bounding boxes depending on the dataset's characteristics. If the dataset has full-body images, the body bounding boxes are extracted. On the other hand, the hand bounding boxes are extracted from the only-hand dataset. After that, the detected bounding boxes will be forwarded to HRNet [18], a CNN-based model, to determine the key points normalized in the processing module. In addition, with the whole body dataset, the hand bounding boxes are evaluated from the farthest keypoint to the left, the farthest keypoint to the right, the farthest keypoint on the top, and the farthest keypoint to the bottom in the processing module. The hand gestures are identified in the classification module that uses keypoints and hand bounding boxes as inputs. The details of each module are described in Sections 3.1–3.3.



(a) Face keypoint



(b) Hand keypoints

Fig. 2. Keypoints.

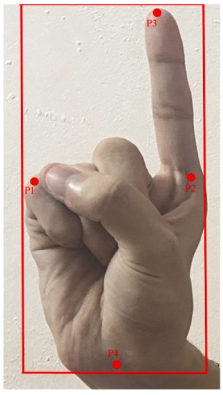


Fig. 3. WH hand bounding box.

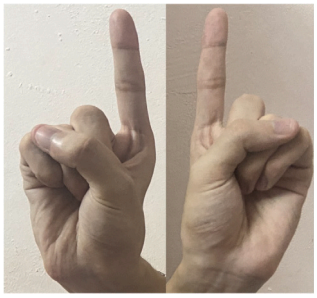


Fig. 4. HBB horizontal stacked.

3.1. Feature extraction module

The feature extraction module employs MMDetection [20] to detect the whole body bounding boxes on the datasets which contain images of the whole or upper half-human body (WH datasets). On the other hand, only hand bounding boxes are detected on datasets with images of close hands and no human body (OH datasets). The extracted bounding boxes are employed with Deep High-Resolution network [18], specifically HRNet_w48 with input size 384 × 288 and 63.6 million parameters, to determine the whole body keypoints on the WH datasets and only hand keypoints on OH datasets.

With whole-body keypoints from the COCO Whole-body dataset [7], HRNet gives us 133 keypoints, including 21 keypoints for the left hand, 21 keypoints for the right hand, 68 keypoints for the face, 17 keypoints for the body, and 6 keypoints for feet. However, our proposed system only used 42 hand keypoints and 1 face keypoint on the nose tip because we found that other keypoints do not bring important information about hand gestures. In addition, hand gestures on some datasets are also interested in the relative position between the face and hands, such as touch cheek and touch head gestures. Therefore, we use the nose keypoint to determine head position to generalize in most HGR tasks with whole or half-human body data. These keypoints are shown in Fig. 2.

Depending on one hand or both hands on OH datasets, 21 or 42 extracted keypoints are used. Fig. 2 illustrated these keypoints.

3.2. Processing module

Keypoints normalization - All extracted keypoints are normalized as [25] because the scale of the human body in different frames is different. They would be in a similar distribution as the original keypoints but the values in the range (−1, 1). Firstly, the center of keypoints is calculated as

$$\begin{aligned} Center_x &= \sum_{i=1}^{n} \frac{x_i}{n} \\ Center_y &= \sum_{i=1}^{n} \frac{y_i}{n} \end{aligned} \tag{1}$$

where $(Center_x, Center_y)$ is the central coordinate of all keypoints and n is the number of keypoints used. Secondly, the coordinate (x_i, y_i) of keypoint P_i is normalized as

$$\begin{aligned} x &= \frac{x_i - Center_x}{l} \\ y &= \frac{y_i - Center_y}{l} \end{aligned} \tag{2}$$

where l is the quantity used to normalize the value of each coordinate into range (−1,1). For WH datasets, l is calculated as Eq. (3) while l is calculated as Eq. (4) for OH datasets.

$$\begin{aligned} l = &max(dis(P_1, P_2), dis(P_1, P_{23})) \\ &+ max(dis(P_2, P_{i,i\in\{3..22\}}), dis(P_{23}, P_{j,j\in\{24..43\}})) \end{aligned} \tag{3}$$

$$l = max(max(dis(P_2, P_{i,i\in\{3..22\}})), max(dis(P_{23}, P_{i,i\in\{24..43\}}))) \tag{4}$$

where $dis(P_i, P_j)$ is the euclidean distance between two keypoints P_i and P_j.

Table 1 shows the explanation of the symbols used in the above equations.

Table 1
Explanation of the symbols.

Symbols	Explanation
$(Center_x, Center_y)$	central coordinate of all keypoints
(x_i, y_i)	coordinate of keypoints P_i
n	number of keypoints
dis(P_i, P_j)	euclidean distance between two keypoints P_i and P_j
l	quantity used to normalize the value of each coordinate into range (−1,1)

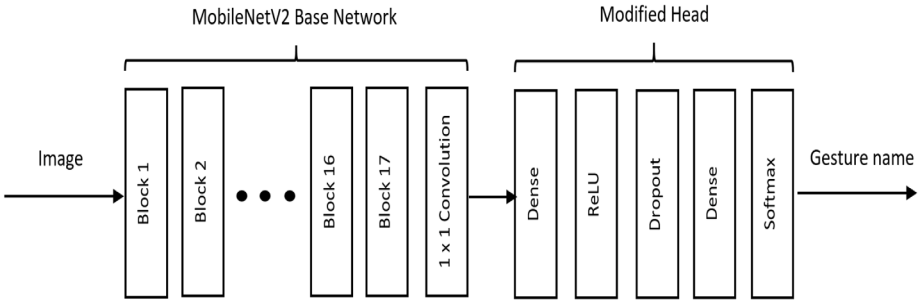


Fig. 5. Fine-tuned MobileNetV2.

Hand Bounding box (BB) - HBB is also used as the input of our proposed architecture. As we mentioned, pose estimation with the top-down method needs bounding boxes as input for keypoint extraction.

On OH datasets, all images contain human hands without bodies, so only HBB is identified to extract hand keypoints. The size of the HBB will be kept without expanding on four sides because the distances between the camera and hands are short. In this situation, the detected HBB is good enough for recognition. On the other hand, all images of WH datasets contain the whole human body, so the human body bounding box is employed to extract whole body keypoints. However, the HBB can also be obtained with WH datasets by using four points, P1, P2, P3, and P4, on hands, where P1 is the farthest point to the left, P2 is the farthest point to the right, P3 is the farthest point on the top, and P4 is the farthest point to the bottom. In this situation, each dataset's WH hand bounding boxes are expanded for 10% or 20% on four sides to limit the loss part of hands. An overview of the WH hand bounding box is shown in Fig. 3.

The collected keypoints and HBB will be used to train the models for recognition.

3.3. Classification module

3.3.1. Method 1: fine-tuned MobileNetV2 + ImageNet pre-trained approach

This approach uses only HBB images as the input for fine-tuned MobileNetV2. If two hand bounding boxes are detected in one image, these two bounding boxes will be resized to 112 × 224 and then horizontal stacked as seen in Fig. 4 to have size 224 × 224. Otherwise, if only one hand bounding box exists in the image, the detected HBB is resized to 224 × 224.

The architecture of MobileNetV2 used to train HBB is customized. Two fully-connected layers replace the last classifier module (Head) to reduce the output units from the previous layer down to 256 and from 256 down to the numbers of output units corresponding to each dataset. Finally, a Softmax classifier is added for multi-class classification. A dropout layer is applied after the first fully-connected layer to reduce overfitting. In addition, the model is trained by fine-tuning an ImageNet pre-trained model on the three datasets. The proposed architecture is shown in Fig. 5.

3.3.2. Method 2: Fully-connected neural network approach (FC)

The keypoints may contain the most necessary information about the gestures. Thus, the keypoints are fed to a fully connected neural network to recognize the hand gesture. As seen in Fig. 6, this approach uses the fully-connected layers with Sigmoid activation functions in the first layers. Because, according to our experiments, Sigmoid activation gives better results than the others. Besides, dropout layers are also added to reduce overfitting. The Softmax activation function is used in the last layer for multi-class classification. Factor N implies that FC has some serial blocks consisting of fully connected, activation, and dropout layers. The value of N is changed on different datasets. The shape of the FC input is equal to the number of keypoints used per sample.

3.3.3. Two-pipeline approach

The use of pose estimation can be used for hand gesture recognition. However, the keypoints are not good enough to recognize the hand gesture in more difficult conditions. To overcome this issue, this manuscript presents a two-pipeline network to combine the features from both keypoints and HBB images. An overview of the network is shown in Fig. 7. As can be seen, there are two parallel pipes in this architecture called keypoints pipeline and hand bounding box pipeline (HB pipeline). The extracted keypoints and HBB are both used to be the inputs of this architecture. This approach has two different method called "MobileNetV2 + FC" and "CNN +FC", respectively.

Method 3: MobileNetV2 + FC. The keypoints pipeline is similar to FC in Section 3.3.2, but Sigmoid activation functions are replaced by ReLU functions based on our experimental results with different activation functions in this architecture. Simultaneously, the pre-trained MobileNetV2 with ImageNet is used to extract the features from HBB in the HB pipeline. Our MobileNetV2 architecture is similar to Section 3.3.1, but the units of fully-connected layers in *Modified Head* are changed to be more suitable for each dataset. Besides, the Softmax classifier of MobileNetV2 is replaced by ReLU activation. The outputs of the two pipes are concatenated together to predict hand gestures from both keypoints and hand bounding boxes.

Method 4: Convolution neural network(CNN) + FC. This architecture is similar to the architecture presented in method 3, but a conventional convolution neural network replaces MobileNetV2 to decrease the number of parameters in the HB pipeline. Three convolution blocks are used in this CNN architecture, with the number of filters for each block being 16, 32, and 64, respectively. The kernel size of these blocks is 5 × 5 or 7 × 7 to suit each dataset. All blocks have

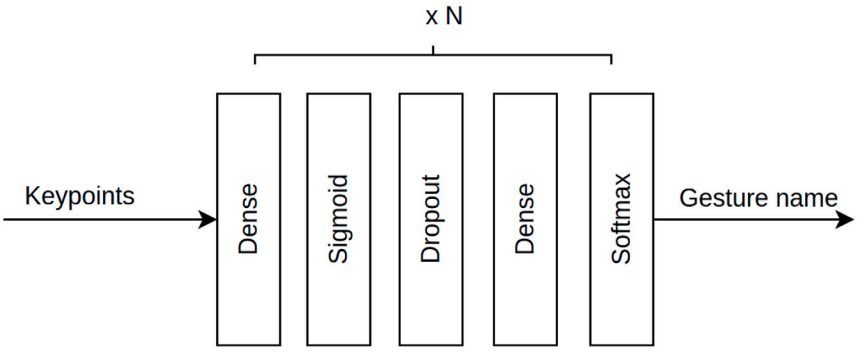


Fig. 6. Structure of the fully-connected neural network.

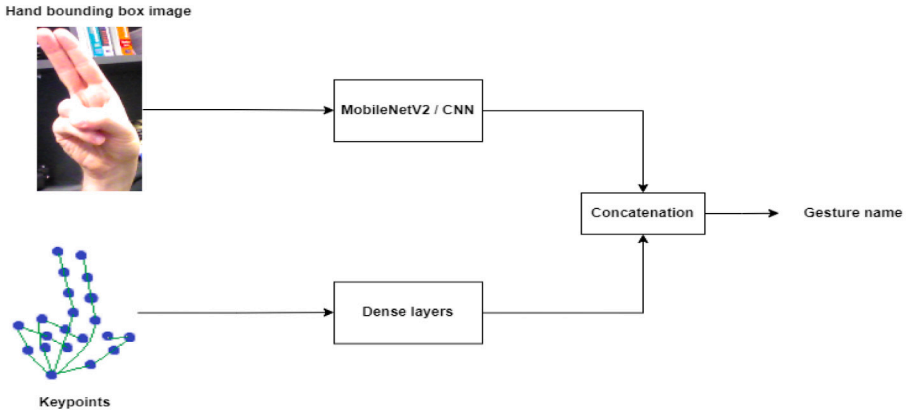


Fig. 7. Structure of the two-pipeline neural network.

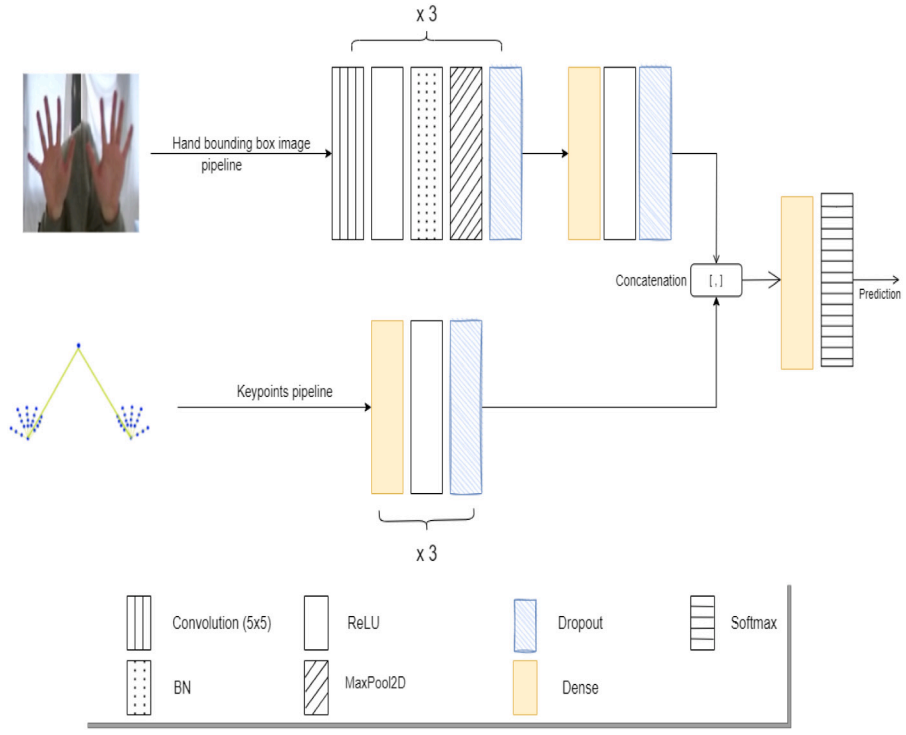


Fig. 8. Structure of the CNN + FC two-pipeline.

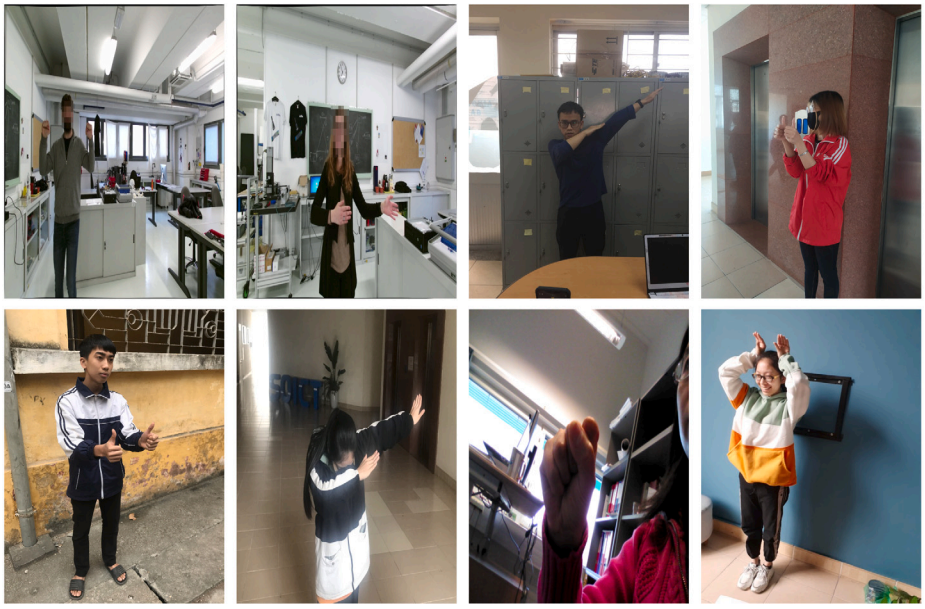


Fig. 9. Examples of images used in test set and validation set.

batch normalization to increase the convergence speed. Dropout layers with the values 0.2, 0.3, and 0.5 are applied to prevent overfitting. All image sizes are reduced from 224 × 224 to 128 × 128 for the input of the HB pipeline. The keypoints pipeline and the concatenation between the outputs of two pipelines are similar to the method 3 architecture. The outputs are fed into a Softmax classifier to predict gestures. The overview of this architecture is illustrated in Fig. 8.

4. Experiments

4.1. Dataset

4.1.1. Hand gestures dataset

To our knowledge, there are few datasets about static human hand gestures today. For example, FreiHAND dataset [26] or Rendered Handpose dataset [27] are hand datasets, but they focus on hand pose estimation, so there is no specific gesture in each image. Therefore, they are difficult to train gesture recognition. In this research, we evaluate our methods on three datasets: HANDS [8], OUHANDS [9] and SHAPE dataset. Among these, HANDS and SHAPE are the WH datasets, and OUHANDS is the OH dataset that we mentioned in Section 3.1

HANDS, a dataset for human–robot interaction, contains both Depth and RGB frames with a full-body or half-body. There are a total of fifteen static two-hand gestures. With five subjects, including three males and two females, and 2400 RGB frames per subject, this dataset has a total of 12000 frames which are collected on different backgrounds.

SHAPE, a dataset for static hand gestures, has twenty subjects consisting of ten male subjects and ten female subjects. All the people perform 32 gestures in some different backgrounds with half-body or full-body. There are over 30000 images with front, top, left, and right camera angles per subject.

OUHANDS, a dataset for hand detection and hand pose recognition, has ten hand gestures for each individual with a total of 3000 images. The images were collected in different backgrounds, subjects, and light conditions. All the hands are located near the camera. In this research, we also want to compare our methods with others by evaluating this dataset because it was also investigated in previous studies.

4.1.2. Data preparing

As mentioned, we evaluated our methods with three datasets, including HANDS, SHAPE, and OUHANDS. For datasets that were explicitly divided into smaller sets for training, validating, and testing by the original authors, we would use those sets without any further modification to compare our results with other studies using similar data sets. Furthermore, for datasets that were not divided by the original authors, we evaluated them using the K-fold validation method.

Because HANDS was not divided by the original authors, this research used the K-Fold validation method to validate this dataset. This dataset has five subjects; we used four subjects for training and the other for validating each time. Therefore, the number of images in the training and validation sets at each time was 9600 and 2400, respectively.

The SHAPE dataset was divided by its author. In this situation, 25678 images were used for the training set, 3621 images were considered the validation set, and 8006 images were reserved for the test set. The test set with 8006 images was again divided into two smaller sets called *Test-Random* with 2855 images and *Test-Final* with 5151 images.

The OUHANDS dataset was also divided into three sets by the original authors. There were 1600 images for training, 400 for validation, and 1000 for testing. The test set was also used by other studies that used to compare with our proposed architecture.

Fig. 9 shows examples of data used in the test and validation sets.

4.2. Experimental setup

We used the Pytorch framework in this study. As explained in Section 3.2, we trained data with several architectures. Based on our experiments, the Adam optimizer algorithm [28] gave better results than the others. Hence, we adopted the Adam algorithm to optimize cross-entropy loss, which was our cost function for experiments with all datasets. For HANDS, SHAPE and OUHANDS datasets, we used respectively the following values for hyper-parameters as shown in Tables 2–4: batch-size, learning rate, number of epochs and image size. Our experiments investigated each learning rate to identify the most suitable value for each method and dataset. Furthermore, the *Epochs* hyperparameter was the number of epochs that we saw the model converged. Our example of DataLoader and four architectures for

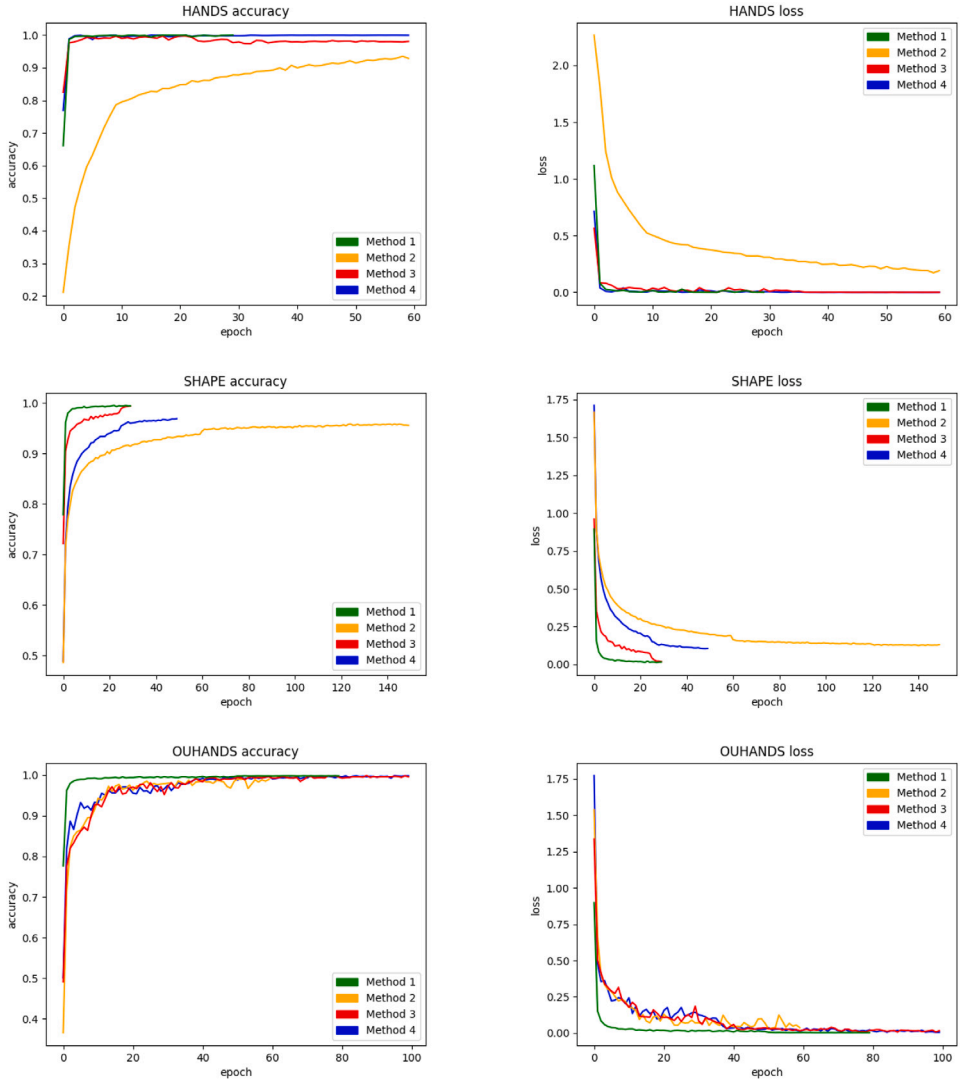


Fig. 10. Training processes (loss and accuracy versus epoch) of each dataset. Each graph simultaneously shows all four processes of the four methods.

Table 2
HANDS hyperparameters.

Method	Batch size	Initial learning rate	Epochs	Image size	Keypoints size
Method 1 (MobileNetV2)	32	0.0001	30	224 × 224	–
Method 2 (FC)	32	0.001	60	–	86
Method 3 (MobileNetV2 + FC)	32	0.0001	60	224 × 224	86
Method 4(CNN + FC)	32	0.001	60	128 × 128	86

Table 3
SHAPE hyperparameters.

Method	Batch size	Initial learning rate	Epochs	Image size	Keypoints size
Method 1 (MobileNetV2)	32	0.0001	30	224 × 224	–
Method 2 (FC)	32	0.001	150	–	86
Method 3 (MobileNetV2 + FC)	32	0.001	30	224 × 224	86
Method 4 (CNN + FC)	32	0.01	50	128 × 128	86

training the SHAPE dataset is available at https://github.com/cvhust/Keypoints_CoEx.

The proposed networks are trained on Nvidia GTX 1080 Ti 11 GB Turbo, double CPU Xeon X5650 with 12 cores and 24 threads, and Google Colab with the standard version.

4.3. Experiments on recognition

Tables 5–7 show the experimental results with HANDS, SHAPE and OUHANDS datasets. We also show our training processes on three datasets in Fig. 10.

Table 5 shows that Method 1 and Method 3 gave the best recognition result on HANDS. Method 4 obtained a good result, with the

Table 4
OUHAND hyperparameters.

Method	Batch size	Initial learning rate	Epochs	Image size	Keypoints size
Method 1 (MobileNetV2)	32	0.0001	80	224 × 224	–
Method 2 (FC)	32	0.01	60	–	42
Method 3 (MobileNetV2 + FC)	16	0.01	100	224 × 224	42
Method 4 (CNN + FC)	16	0.01	100	128 × 128	42

Table 5
HANDS recognition results.

Method	Test accuracy	Test precision	Test recall	Parameters	Floating-point operations per second
Method 1	0.98	0.97	0.98	2.56 *millions*	313.62 *millions*
Method 2	0.89	0.87	0.88	0.02 *millions*	0.02 *millions*
Method 3	0.98	0.97	0.98	2.71 *millions*	313.46 *millions*
Method 4	0.91	0.90	0.90	0.22 *millions*	95.93 *millions*

Table 6
SHAPE recognition results.

Method	Test accuracy		Test recall		Test precision		Parameters	FLOPs
	Test-Final	Test-Random	Test-Final	Test-Random	Test-Final	Test-Random		
Method 1	0.95	0.96	0.95	0.96	0.96	0.96	2.56 *millions*	313.62 *millions*
Method 2	0.94	0.94	0.94	0.94	0.94	0.94	0.03 *millions*	0.02 *millions*
Method 3	0.97	0.98	0.96	0.98	0.96	0.98	2.58 *millions*	313.64 *millions*
Method 4	0.95	0.96	0.94	0.96	0.94	0.95	0.23 *millions*	95.93 *millions*

Table 7
OUHANDS recognition results and comparison.

Method	Test accuracy	Test recall	Test precision	Parameters	Floating-point operations per second
Method 1	0.94	0.94	0.95	2.56 *millions*	313.61 *millions*
Method 2	0.93	0.93	0.93	0.01 *millions*	0.01 *millions*
Method 3	0.94	0.93	0.94	2.57 *millions*	313.63 *millions*
Method 4	0.94	0.94	0.94	0.22 *millions*	157.36 *millions*
HGR-Net [13]	0.88	0.88	0.89	–	–
Fit-Hand [29]	0.65	–	–	–	–

number of parameters being less than Method 3 about ten times. The recognition result of method 2 using only keypoints pipeline in this situation gave a not too good result. However, when combining the keypoints pipeline and bounding boxes image pipeline (Method 3 and Method 4), the performance of the models was significantly improved.

Table 6 demonstrated that Method 3 had the best results on both test sets. The others also achieved high accuracy.

Table 7 shows our recognition results and compares them with some other architectures on the OUHANDS test set. The first four rows present our recognition results. The last two rows are the accuracy of previous studies, including HGR-Net [13] and Fit-Hand [29]. It showed that our system outperformed previous studies.

The above results showed that our two-pipeline architecture with a top-down pose estimation method reached stable high results in the HGR task regardless of the long or short distance between the camera and human hands or different light conditions and backgrounds. The reason is that our architecture can learn to perform information from both hand bounding box images and keypoints and then combine them to predict hand gestures better. This approach can be useful when the extracted keypoints or hand bounding box images are not good enough for gesture recognition in difficult conditions. In addition, the number of parameters in the two-pipeline architecture is not too large. Besides, simple FC architecture may also reach high accuracy because the extracted keypoints are quite informative enough for classification, so FC with only fully-connected layers can well learn from only the coordinates of the keypoints to give the prediction. It can be concluded that with the high accuracy requirements, the proposed two-pipeline architecture is a suitable option. On the other hand, FC architecture is a suitable option that may have good results with lightweight requirements.

The proposed method gives good results in most cases. However, in some difficult circumstances, it can still be confused. The performance of the proposed model depends on the keypoints and the detected bounding boxes of the hands. Although the models can still predict the gesture in case one of the two streams gives bad results, the performance and certainty of the prediction are still affected. Fig. 11 shows some examples of data and their visualized keypoints that can degrade the performance of the proposed method. Specifically, in some cases, if images are too dark, the human hands are very far from the camera, or the hand postures that the fingers are partially obscured, the extracted keypoints are often inaccurate. This situation leads to the keypoints of some gestures being similar to other gestures, and recognition results can be confused. Those image types can be seen as images: I1, I2, I3, and I4 in Fig. 11. Another case that can also reduce the recognition performance of the model is when the gestures are acted too similarly, although their labels are different. Therefore, in some camera angles, these actions are difficult to distinguish. This image type can be seen as images: I5 and I6 in Fig. 11.

The experiment results in this research demonstrated that our proposed system worked well with diverse datasets. In addition, the processing time of the system was quite short. The total time to pre-process an image and recognize its gesture was about 0.22(s) on GPU Nvidia GTX 1080. The training speed with batch size 32 of Method 1, Method 2, Method 3, and Method 4 were respectively about 98 (s/epoch), 6 (s/epoch), 98 (s/epoch), and 16 (s/epoch), where s/epoch was the seconds per epoch. Furthermore, the space complexity with the same batch 32 in the training process of four methods was approximately 1 gigabyte, 0.8 gigabyte, 3.5 gigabyte, and 3.5 gigabyte, respectively.

5. Conclusion

This manuscript presents four classification architectures for static hand gesture recognition with data processed by top-down pose estimation using HRNet. This research also recommends what keypoints are

Fig. 11. Examples of images that can degrade the performance of the proposed method.

needed and how to normalize them for this task. Among four architectures, the two-pipeline architecture can learn the features combined from both hand bounding boxes and keypoints. The experiments were conducted on three datasets called HANDS, SHAPE, and OUHANDS which contain images under various conditions, including background, light condition, distance, human sex, and age, to investigate different solutions. Our experiments on those datasets showed that the proposed architecture might obtain high accuracy with hand gesture recognition tasks. To our knowledge, our results are the best results on HANDS, OUHANDS, and SHAPE datasets until the present time.

In our future work, we hope to use our methods with the data processed by bottom-up pose estimation (HigherHRNet) [30] to improve the results further. Besides, it is necessary to investigate more lightweight models to use this system in real-time applications and mobile devices, collect more data to train a better model, and use other models in the hand bounding box pipeline.

CRediT authorship contribution statement

Tuan Linh Dang: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Sy Dat Tran:** Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Thuy Hang Nguyen:** Software, Data curation, Writing – original draft, Visualization. **Suntae Kim:** Conceptualization, Methodology, Validation, Supervision, Project administration. **Nicolas Monet:** Conceptualization, Methodology, Writing – review & editing, Validation, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the cooperation between Hanoi University of Science and Technology and Naver Corporation.

References

[1] Isaacs J, Foo S. Hand pose estimation for american sign language recognition. In: Thirty-sixth southeastern symposium on system theory, 2004. proceedings of the. IEEE; 2004, p. 132–6.
[2] Phyo AS, Fukuda H, Lam A, Kobayashi Y, Kuno Y. A human-robot interaction system based on calling hand gestures. In: International conference on intelligent computing. Springer; 2019, p. 43–52.
[3] Dardas NH, Alhaj M. Hand gesture interaction with a 3D virtual environment. Res Bull Jordan ACM 2011;2(3):86–94.
[4] Ge L, Cai Y, Weng J, Yuan J. Hand pointnet: 3d hand pose estimation using point sets. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 8417–26.
[5] Hong C, Yu J, Zhang J, Jin X, Lee K-H. Multimodal face-pose estimation with multitask manifold deep learning. IEEE Trans Ind Inf 2018;15(7):3952–61.
[6] Murphy-Chutorian E, Trivedi MM. Head pose estimation in computer vision: A survey. IEEE Trans Pattern Anal Mach Intell 2008;31(4):607–26.
[7] Jin S, Xu L, Xu J, Wang C, Liu W, Qian C, et al. Whole-body human pose estimation in the wild. In: European conference on computer vision. Springer; 2020, p. 196–214.
[8] Nuzzi C, Pasinetti S, Pagani R, Coffetti G, Sansoni G. HANDS: an RGB-D dataset of static hand-gestures for human-robot interaction. Data Brief 2021;35:106791.
[9] Matilainen M, Sangi P, Holappa J, Silvén O. OUHANDS database for hand detection and pose recognition. In: 2016 sixth international conference on image processing theory, tools and applications. IEEE; 2016, p. 1–5.
[10] SHAPE dataset. 2021, https://users.soict.hust.edu.vn/linhdt/dataset/.
[11] Pansare JR, Gawande SH, Ingle M. Real-time static hand gesture recognition for American sign language (ASL) in complex background. 2012.
[12] Bretzner L, Laptev I, Lindeberg T. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In: Proceedings of fifth IEEE international conference on automatic face gesture recognition. IEEE; 2002, p. 423–8.
[13] Dadashzadeh A, Targhi AT, Tahmasbi M, Mirmehdi M. HGR-net: a fusion network for hand gesture segmentation and recognition. IET Comput Vis 2019;13(8):700–7.
[14] Liang C, Song Y, Zhang Y. Hand gesture recognition using view projection from point cloud. In: 2016 IEEE international conference on image processing. IEEE; 2016, p. 4413–7.
[15] Moin A, Zhou A, Rahimi A, Menon A, Benatti S, Alexandrov G, et al. A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. Nat Electron 2021;4(1):54–63.
[16] Ning G, Liu P, Fan X, Zhang C. A top-down approach to articulated human pose estimation and tracking. In: Proceedings of the european conference on computer vision (ECCV) workshops. 2018.
[17] Luvizon DC, Picard D, Tabia H. 2D/3d pose estimation and action recognition using multitask deep learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 5137–46.
[18] Sun K, Xiao B, Liu D, Wang J. Deep high-resolution representation learning for human pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 5693–703.
[19] Wang J, Sun K, Cheng T, Jiang B, Deng C, Zhao Y, et al. Deep high-resolution representation learning for visual recognition. IEEE Trans Pattern Anal Mach Intell 2020.
[20] Chen K, Wang J, Pang J, Cao Y, Xiong Y, Li X, et al. Mmdetection: Open mmlab detection toolbox and benchmark. 2019, arXiv preprint arXiv:1906.07155.
[21] Saxen F, Werner P, Handrich S, Othman E, Dinges L, Al-Hamadi A. Face attribute detection with mobilenetv2 and nasnet-mobile. In: 2019 11th international symposium on image and signal processing and analysis. IEEE; 2019, p. 176–80.
[22] Baumgartl H, Sauter D, Schenk C, Atik C, Buettner R. Vision-based hand gesture recognition for human-computer interaction using MobileNetV2. In: 2021 IEEE 45th annual computers, software, and applications conference. IEEE; 2021, p. 1667–74.
[23] Sagayam KM, et al. CNN-based mask detection system using opencv and MobileNetV2. In: 2021 3rd international conference on signal processing and communication. IEEE; 2021, p. 115–9.
[24] Ma W, Lu J. An equivalence of fully connected layer and convolutional layer. 2017, arXiv preprint arXiv:1712.01252.
[25] Pismenskova M, Balabaeva O, Voronin V, Fedosov V. Classification of a two-dimensional pose using a human skeleton. MATEC Web Conf 2017;132:05016. http://dx.doi.org/10.1051/matecconf/201713205016.
[26] Zimmermann C, Ceylan D, Yang J, Russell B, Argus M, Brox T. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 813–22.

[27] Zimmermann C, Brox T. Learning to estimate 3d hand pose from single rgb images. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 4903–11.

[28] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, arXiv preprint arXiv:1412.6980.

[29] Verma M, Gupta A, et al. One for all: An end-to-end compact solution for hand gesture recognition. 2021, arXiv preprint arXiv:2105.07143.

[30] Cheng B, Xiao B, Wang J, Shi H, Huang TS, Zhang L. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 5386–95.