Comparison of Two Approaches to Task-specific Real-Time Hand Pose Estimation

Guram Chaganava[†] **and David Kakulia**[†],
Guram.chaganava037@ens.tsu.edu.ge
[†]Ivane Javakhishvili Tbilisi State University, Tbilisi, Georgia

Abstract
Real-time hand pose estimation in an image plays an important role in systems that require human-computer interaction (HCI). In some cases, a task requires hand pose estimation, not in any, but images with specific content. For example, such a task may require hand pose estimation in images showing one person speaking sign language near the camera. The goal of the study presented in this paper is to experimentally test the assumption that, for the aforementioned specific tasks it will be better than the standard approach to perform hand pose estimation directly in the original image, without hand detection. This approach can result in higher speed and nearly the same accuracy of hand pose estimation as in the case of the standard approach. To determine the advantage of the direct approach for specific tasks, it is necessary to compare the methods in terms of accuracy and speed. For this, a comparative analysis of the standard and direct approaches is carried out. The efficiency coefficients of the methods are quantitatively evaluated to find the optimum between accuracy and speed. It is also examined how the accuracy of hand pose estimation depends on the content of the dataset used to build such a system. As a result, the direct approach proves to be more efficient when using a dataset consisting of images with specific content.

Key words:
Hand Pose Estimation, Keypoint, Keypoint Detection, Hand Detection.

1. Introduction

Hand pose estimation refers to the process of modelling a human hand as a set of some objects. It is a common practice to model a hand as a system of keypoints. A keypoint is a specific place on the human body. Using coordinates of keypoints, a position of a human body can be restored in space. Examples of hand keypoints include the joint of a wrist, finger joints, etc. The process of determining the coordinates of keypoints is called keypoint detection. Keypoint detection is one of the most popular tasks of computer vision. It is used in tasks, such as virtual and augmented reality (VR/AR) systems[1], interactive games [2], gesture recognition[3], action recognition[4], computer-aided design (CAD)[5], sign language recognition[6], etc.

The publications [7], [8] present state-of-the-art techniques for 2D and 3D hand pose estimation in images. As follows from these articles, the vast majority of researchers do not perform hand pose estimation directly in full images. Instead, at first, they determine the location of the hand in an image, crop it and then detect keypoints in the cropped image. As a result, hand pose estimation problem is divided into two tasks:

(i) **Hand detection**. It implies determining whether the hand is shown in a photograph, and if so, how many. If at least one hand is presented in an image, its location must be found.
(ii) **Hand keypoint detection**. After hand detection, each found hand is cropped from the image. Pre-processing of the cropped image is performed, such as resizing, normalizing, etc. The resulting image is fed to a keypoint detector.

In this case, the problem relatively simplifies since it is divided into two tasks of less complexity. As a result, this approach makes it possible to achieve a high degree of generalization - estimating hand pose with high accuracy in images with any content. Because of this, the use of this technique for hand pose estimation has become the standard. On the other hand, this approach is more time consuming as it is necessary to solve two different tasks. It is important that in both cases the detection algorithm is optimised so that the detection time is as short as possible. Detection time, along with accuracy, is a key characteristic of real-time systems. Choosing the optimal algorithms for both tasks, resulting in the high accuracy and speed of detection, can sometimes be problematic.

Sometimes a task requires hand pose estimation not in any, but specific conditions. For example, interactive computer games may require hand pose estimation in photographs that depict a person sitting in front of the computer and holding his/her hand in a specific position. A similar situation occurred in the case of the task that required real-time hand pose estimation in photographs showing one person speaking sign language near the camera. Since the signer stands close to the camera, he/she will be displayed in most of the photograph. The hand/hands will also occupy a large area in the image. According to the purpose of the task, the hand pose estimation should be done only in the photographs of this content category.

The assumption was made that the hand pose estimation would be performed faster if keypoints were searched not in cropped, but directly in the original image. In terms of accuracy, since the estimation of hand pose is

Manuscript received October 5, 2021
Manuscript revised October 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.10.32

only required only in the category of photographs described above, the accuracy of detection in original images could be close to the accuracy of detection in cropped images. Experimental examination of these assumptions is the goal of the paper. To test the advantage of a direct hand pose estimation over a standard approach for specific tasks, it is necessary to compare systems to each other that implement both approaches. For these hand pose estimation systems to be comparable, they must be built based on the same principles. The systems should be compared in terms of accuracy and speed. Techniques used for building hand pose estimation systems, a method for comparing approaches and the experiment results are discussed in the paper.

2. Methodology

Comparison of the two approaches described above requires hand pose estimation systems that implement both of them. From Fig. 1 and 2 it is clear what parts both systems consist of. Building a standard hand pose estimation system requires a hand detector and a keypoint detector able to find keypoints in cropped images. For direct hand pose estimation a keypoint detector is needed which will perform detection in original images. For the keypoint detectors to be comparable, their operation must be based on a same working principle.

Fig. 1. The standard, two-stage hand pose estimation workflow

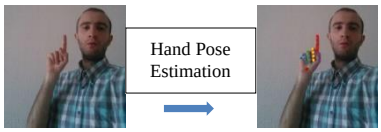


Fig. 2. The direct, one-stage hand pose estimation workflow

The two approaches of hand pose estimation should be compared according to accuracy and speed. Since the standard approach performs hand pose estimation in two main steps, let's assign it as HPE2 (Hand Pose Estimation 2 step) and the direct approach as HPE1 (Hand Pose Estimation 1 step). The following abbreviations are also used in the paper: KDO (Keypoint Detection in Original images), KDC (Keypoint Detection in Cropped images), HD (Hand Detection). An accuracy is denoted as A, time as T. For example, hand detection accuracy is denoted as A_{HD}, and detection time as T_{HD}.

The accuracies A_{KDO} and A_{HPE1} are the same since hand pose estimation in original images is done in only one step. A_{HPE2} depends on A_{HD} and A_{KDC}. T_{KDO} and T_{HPE1} are also the same. T_{HPE2} is the sum:

$$T_{HPE2} = T_{HD} + T_{KDC} \quad (1)$$

2.1 Keypoint detection

Keypoint detectors can be built using a deep learning approach which is quite successful in such tasks. To minimize detection time, it is better to choose a deep learning model with relatively small size such as Mobilenet[9], [10].

To train the model for detecting keypoints in original and cropped images, a dataset is required containing images and corresponding targets – x and y coordinates of each keypoint in each image. The dataset must contain photographs that correspond to the given task. Let's denote the dataset as $\{P, K\}$. P is the set of images and K is the set of corresponding targets. The resolution of images of set P is equal to (w, h).

$$\{P, K\} = \{(p_1, k_1) \cup (p_2, k_2) \cup ... (p_n, k_n)\} = \bigcup_{i=1}^{n} (p_i, k_i) \quad (2)$$

$\{P, K\}$ is a set of pairs (p_i, k_i), where p_i is the corresponding matrix of an image with index i, and k_i is an array containing the coordinates of the keypoints.

$\{P, K\}$ dataset can be used for training the model to detect keypoint in original images. In case of detection in cropped images, a dataset is required containing images of hands and the corresponding targets. Let's denote this dataset as $\{P', K'\}$. P' is the set of submatrices derived from p_i matrices. Each submatrix corresponds to a part of a photograph that shows a human hand. K' is a set of k_i' arrays of coordinates of keypoints.

To get the p_i' submatrix, part of the p_i image must be found and cropped where a hand palm is shown. k_i array can be used to find the location of the hand palm. Boundaries of the hand palm in the p_i image can be determined by the minimum and maximum values of coordinates of keypoints along the x and y axes. These points form a bounding box of the hand palm. The bounding box will intersect the outer keypoints. So, part of the hand palm may be left outside the bounding box. Therefore, the size of the bounding box is increased along the x and y axes in proportion to its width and height, respectively. The part of the image p_i, enclosed by the bounding box, is cropped. Cropped images vary in size.

However, the neural network used for detection has a fixed input size. Therefore, all images must have the same (w', h') resolution. w' and h' can be equal to each other $(w' = h')$, similar to w and h. All images that have at least one size (width or height) greater than w', are resized so that their largest size is w'. The second size of the resulting images is less than w'. Then the images are placed in the centre of a matrix of size (w', h') filled with values equal to 255, which corresponds to white colour. Images smaller than w' in both sizes are placed in the matrix without resizing. As a result, set P' is obtained containing images of size (w', h').

To get the k_i' target, few transformations must be performed on each k_i array. At first, the x and y coordinate of the k_i array must be translated with the coordinates of the start point of the bounding box. Then the coordinates should be scaled so that their x and y values must fall into the intervals: $[0, w']$ and $[0, h']$, respectively. Performing such transformations for each element of the set K will result in a new set of targets K'.

It is expected that the accuracy of keypoint detection in full and cropped images will be depended on the content of a dataset used for training. Consider the case where a dataset includes images of several people, in which a hand is shown in different positions. Using such a dataset, keypoint detection in cropped images can be performed with high accuracy. The cropped images practically do not show the person, the background and the surrounding objects. So, the coordinates of keypoints are mainly determined by the image of a hand palm. Because of this, there is a chance that the training will be successful.

Regarding detection in original images, the detected keypoint coordinates depend on the value of each pixel of the full image. The images of the aforementioned dataset differ significantly from each other in content. Because of this, each new sample of the dataset will lead to significant changes in weight during the training process. Probably, high-accuracy detection will not be possible to achieve in this case.

Let's say the dataset contains not one but many images of different people. In the images of a particular person, the speaker is shown against the same background in almost the same lighting conditions, albeit with different hand positions. In this case, the main difference in the content of the images is the position of a hand. The rest of the details of the images are almost the same. When using such a dataset for training, the weight change will be mainly due to the hand position. Consequently, relatively high detection accuracy can likely be obtained using such a dataset.

To test the aforementioned assumptions experimentally, datasets can be built with different content:

- Dataset 1 - containing images of one person showing a hand in different positions.
- Dataset 2 - containing images of several people showing a hand in one specific position.
- Dataset 3 - containing images of several people showing a hand in different positions.

The purpose of creating these three datasets is not to train a real-world keypoint detector, but to study how the accuracy of keypoint detection in original and cropped images depends on the content of a dataset. Therefore, there is no need to have numerous samples in a dataset.

OpenPose[11] system was used to get the target of every image of each dataset. OpenPose returns x and y coordinates of 21 hand palm keypoints from each image. Detection results were corrected after a visual inspection. Position and brightness augmentation were used to increase the number of samples in the dataset.

2.2 Hand detection

Hand detection involves finding a hand palm location in an image. This location is returned as a bounding box.

A deep learning model can be trained to create a hand detector. This requires a dataset, containing images and the corresponding targets containing coordinates of hand bounding boxes. Such a dataset can be easily obtained from the set $\{P, K\}$. Let's denote the dataset required for hand detection as $\{P, B\}$. B is a set of bounding box coordinates. The coordinates of the bounding box boundaries are easily determined from the set K, as described above. The set $\{P, B\}$ can be used to train Mobilenet for hand detection. The resulting detector can be compared to existing detectors in terms of accuracy and speed.

2.3 Metric

For measuring the keypoint detection accuracy, the Percentage of detected Joints (PDJ)[12] can be used. This is the ratio of the number of correctly detected keypoints to the total number of keypoints in the image. The detection is considered correct if the Euclidean distance between the target and the predicted keypoints is less than or equal to the maximum acceptable error. The length of the diagonal of the bounding box covering the human body in the image is used to determine the value of the maximum acceptable error. Since the detector must find the keypoints of not the entire body, but only the hand palm, the diagonal of the hand bounding box is used to

determine the maximum acceptable error as it is described in [13].

In addition to the keypoint detection accuracy, hand detection accuracy also must be measured. There are many methods to measure object detection accuracy[14]. One of the specific cases of object detection is hand detection. When a task requires detection in images showing one object of one category, as in this case (only hand category), a relatively simple method such as Intersection Over Union (IOU) can be used. It is the ratio of the areas of the intersection and union of ground-truth and predicted bounding boxes. IOU must be calculated for each image and then averaged over the entire dataset.

HPE1 and HPE2 systems, keypoint and hand detectors should be compared to each other according to accuracy and speed. Some quantity can be used for this purpose. Let's call it the efficiency coefficient. If the efficiency coefficient is high, then both accuracy and speed are high, not one of them. It depends on conventional quantities: accuracy and speed coefficients, c^a and c^t, respectively. These coefficients must be constrained with some bounds, for example, it can be the interval [0, 1]. To get the c^a value for each model, the value of accuracy of the given model should be divided by the maximum accuracy of the models:

$$c_i^a = \frac{a_i}{\max\limits_{j=1,...,N} a_j}, i = 1,2,...,N \quad (3)$$

where N represents the number of models among which we want to determine the most efficient one.

For calculating the speed coefficient c^t, it must be taken into account that that speed is inversely proportional to time - longer detection time means lower speed.

To get the speed coefficients, every value of detection time can be substructed from the maximum value of time. However, the coefficient c^t corresponding to the maximum value will be equal to 0, which is undesirable, since it is planned to calculate efficiency coefficients with c^a and c^t. Because of this, each value of time is subtracted not from the maximum, but from some t' number so that $t' > \max\limits_i a_i$. After subtraction the resulting values are normalized.

$$c_i^t = \frac{t' - t_i}{\max\limits_{j=1,...,N}(t' - t_j)}, i = 1,2,...,N \quad (4)$$

The coefficient of efficiency c^e of the given model can be defined as the product of the coefficients of accuracy and speed.

$$c_i^e = c_i^a * c_i^t, i = 1,2,...,N \quad (5)$$

3. Results and Discussion

For the experiment HPE1 and HPE2 systems has been created. The approaches were compared in terms of accuracy and speed.

3.1 Keypoint detection

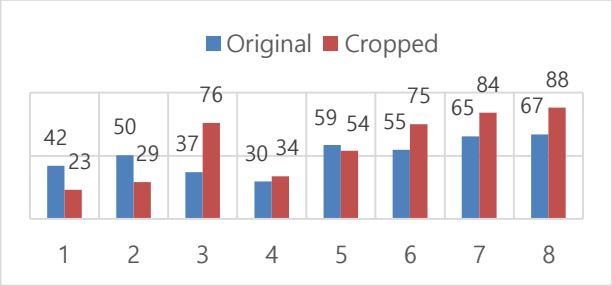
To create a detector, a deep learning model - Mobilenet has been used. Mobilenet, pretrained for feature vector extraction was taken from [15]. There are different versions of Mobilenet, with different depth multipliers and input sizes. The input size must be equal to the resolution of the image being fed to the network. The resolution of the original image is 224x224 while in the case of cropped image it is 128x128. Experimenting using models with different depth multipliers will allow observing the difference between the two approaches on several results. So, keypoint detectors have been created using different modifications of Mobilenet, which are presented on Tab. 1.

Table 1: Versions and depth multipliers of Mobilenet used in the experiment. The parentheses contain a conventional name for the specific Mobilenet, as they are mentioned in the graphs below.

Mobilenet version	**Depth multiplier**			
1	0.25(1)	0.5(2)	0.75(3)	1(4)
2	0.35(5)	0.5(6)	0.75(7)	1(8)

A dense layer with 42 neurons (the number of targets) was added to the last layer of each deep learning model. Hyperbolic tangent was chosen as the activation function of the last layer. Before training, the values of p_i and p_i' matrices were scaled in the range [0,1], k_i and k_i' matrices were scaled in the range [-1, 1]. For the training process next device was used: CPU - Intel Core i7-3632 2.2 GHz, RAM – 6GB.

Three datasets have been used in the experiment. The number of samples in the first two datasets was within a few hundred, and in the third, it exceeded 3000. All deep learning models were trained using these datasets.



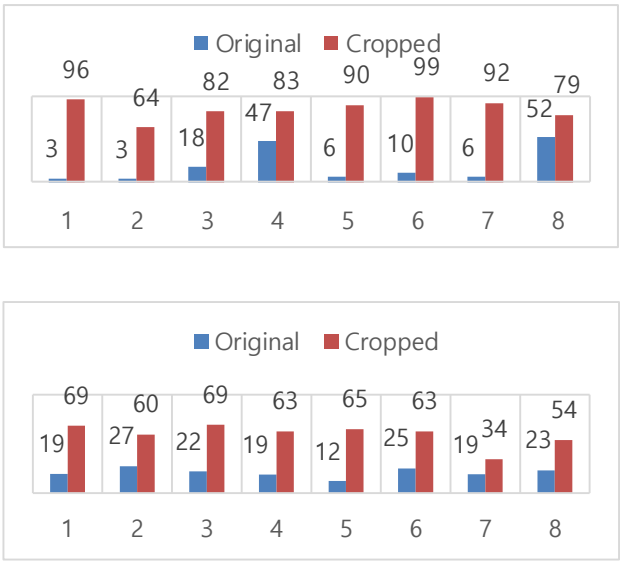


Fig. 3. Accuracy of models trained on Dataset 1(upper), 2 (middle) and 3 (lower) for keypoint detection in original and cropped images. Accuracy is measured on the test set in PDJ percentage. Horizontal axis titles show the Mobilenet conventional names.

The assumption was partially justified stating that HPE1 could have outperformed in the case of training using Dataset1. In the case of only three models, detection in the original image was found to be more accurate than in the cropped image.

In the case of training using Dataset 2, the aforementioned assumption was justified for all models. In each case, preference was given to the detection in the cropped images.

In the case of training using Dataset 3, the preference was given to the method of detection in cropped images. However, the maximum accuracy obtained is not as high. This can be due to a few reasons. The first reason can be the insufficient size of the dataset. Compared to the other two datasets, Dataset 3 consists of images with relatively diverse content. Presumably, the number of samples of Dataset 3 was not enough to learn detection in various situations. The reason for the resulting low accuracy may also be that the number of samples representing each hand pose in the dataset is different. To build a real-world detector, it would be better to use a relatively large dataset, in which the number of samples representing each hand pose will be close to each other.

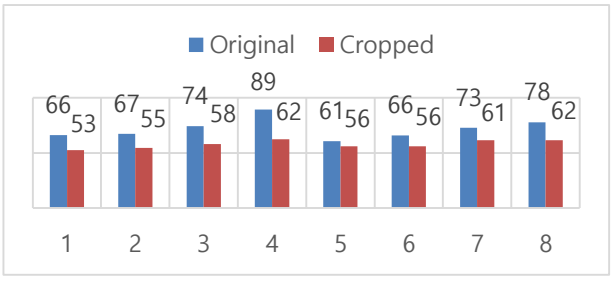


Fig. 4. Keypoint detection times for each model measured in milliseconds. The input size of each model used for detection in original images is 224x224, while the input size of models used for detection in cropped images is 128x128.

The difference between the time it takes to detect keypoints in an original and cropped image is due to the difference between the input sizes of the models. The results show that the time required for detection depends on the depth multiplier for both versions of the Mobilenet. The larger the depth multiplier, the more detection time.

The coefficients of efficiencies were calculated for each model. To calculate the coefficients c^t , it was necessary to determine the value of t' which must be close to the value of maximum detection time. Since the maximum detection time value is 89, so let's t' be 100. The efficient models were determined using the efficiency coefficients. They are listed on Tab. 2.

Table 2: The most effective KDO and KDC models in case of using each dataset.

Dataset	KDO models	KDC models
Dataset 1	v2, 0.35	v2, 1
Dataset 2	v2, 1	v1, 0.25
Dataset 3	v1, 0.5	v1, 0.25

To improve accuracy, each efficient model was retrained with the same dataset, albeit using different loss functions: Mean Squared Error (MSE) and Mean Absolute Error (MAE). In Fig. 5, there are shown the maximum accuracies obtained for every efficient model.

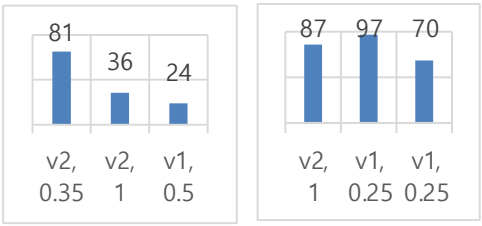


Fig. 5. The accuracy of the most effective KDO (left) and KDC (right) models after retraining. Accuracy is measured in PDJ percentage.

3.2 Hand Detection

Eight Mobilenet models were trained for hand detection using Dataset 3 as it is a more diverse dataset than the other two. The most efficient model turned out to be Mobilenet v1 with a depth multiplier of 0,25 (denoted as 1 in figures). This model was compared to a few existing hand detectors which include: ssd_mobilenet_v1_coco (denoted as 2 in figures) trained on Egohands Dataset[16] by Victor Dibia[17], Different versions of YOLO[18] trained by cansik[19]:

(i) YOLO v3 (denoted as 3 in figures)
(ii) The slim version of YOLO v3 called YOLOv3-tiny (denoted as 4)

(iii) The tiny version of YOLO improved by partial residual networks[20] called Yolov3-Tiny-PRN (denoted as 5)
(iv) YOLOv4 tiny (denoted as 6).

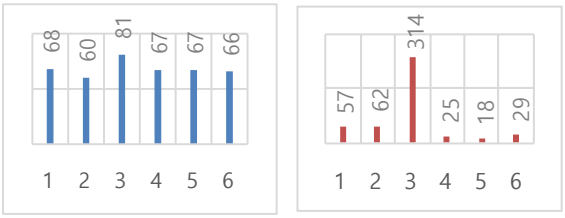


Fig. 6. Hand detection accuracy measured in IOU (left). Hand detection time in ms (right).

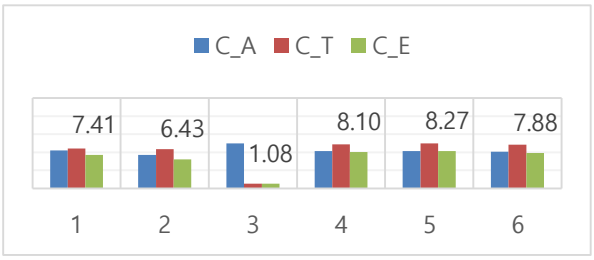


Fig. 7. Efficiency of HD models. Accuracy (denoted as C_A), speed (C_T), and efficiency (C_E/10) coefficients of hand detector models.

As seen from the diagrams, the highest accuracy was obtained with the YOLO v3, although the average detection time was quite long compared to other models. In other cases, the detection accuracies were relatively close to each other. The highest efficiency was observed in the case of Yolov3-Tiny and Yolov3-Tiny-PRN.

3.3 Comparison of approaches

The two approaches to hand pose estimation have been compared using the selected efficient models in terms of accuracy and speed. Hand pose estimation accuracy and time of approach HPE1 are the same as the accuracy and time of keypoint detection in the original images. To measure hand pose estimation accuracy and time of the HPE2 approach, the following steps were performed: an image with the resolution 224x224 has been fed to the most efficient hand detector. T_{HD} time was spent on hand detection. The detector returned the coordinates of the hand bounding box. Using these coordinates the hand image was extracted from the original image. This image was resized to 128x128 and passed to the keypoint detector. It returned the final result - the coordinates of the keypoints. The total time T_{KDC} was spent on the next operations: cutting and resizing the hand image, detecting keypoints. The total time T_{HPE2} spent on estimating the hand pose was calculated by finding the sum of T_{HD} and T_{KDC}.

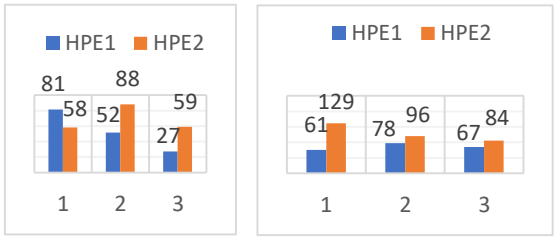


Fig. 8. Accuracy of HPE1 and HPE2 approaches in PDJ percentage (left). Time spent on hand pose estimation by HPE1 and HPE2 approaches in ms (right). Accuracy and time are measured on efficient models selected in the case of each dataset. The dataset names are shown below each figure.

As seen from Fig. 9, the accuracy of the HPE2 approach is significantly lower than the accuracy of the keypoint detection in the cropped images on each dataset. This is caused by defective hand detection in some cases. In terms of speed, less time was spent on hand pose estimation by the HPE1 approach in each case.

To calculate the rate factor c^t, let $t^{'}$ be 150 since that is the closest number to the maximum value of inference time.

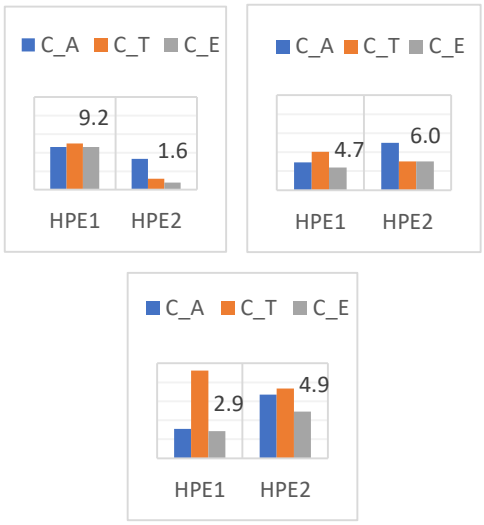
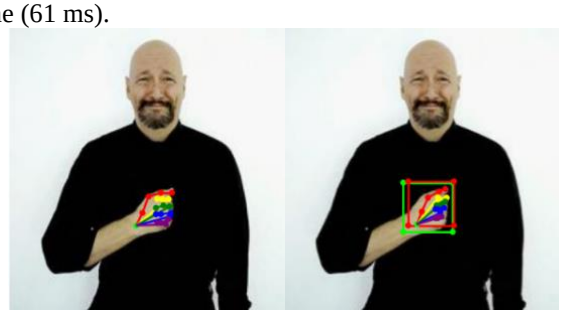


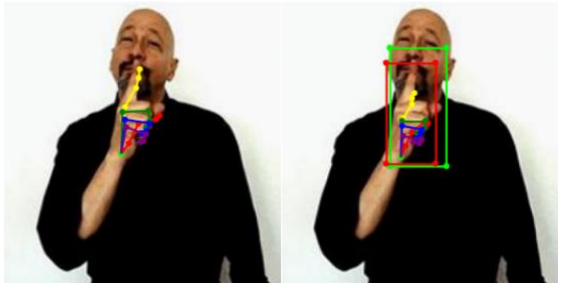
Fig. 9. The coefficients of accuracy (denoted as C_A), speed (C_T) and efficiency (C_E/10) of HPE1 and HPE2 approaches, measured on efficient models selected in the case of each dataset (Dataset 1 – upper left, Dataset 2 – upper right, Dataset 3 – bottom).

From Fig. 10, it can be concluded that the HPE1 approach was more efficient only when using Dataset 1. In this case, there was a significant difference in the coefficients of the efficiencies of the approaches, which was not the case in the other two datasets. This is due to the relatively high accuracy (81%) and short detection

time (61 ms).

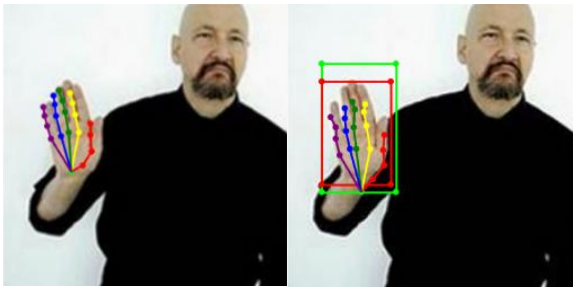


$A_{HPE1} = 100\%$ $A_{HD} = 75,6\%$ $A_{HPE2} = 42,86\%$



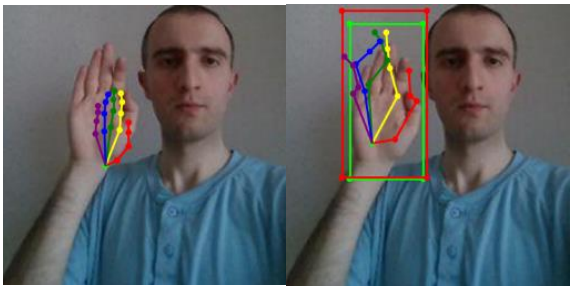
$A_{HPE1} = 95,24\%$ $A_{HD} = 65,38\%$ $A_{HPE2} = 76,19\%$

Fig. 10. Results of direct (left) and standard (right) hand pose estimation in images of Dataset 1. True (green) and predicted (red) bounding boxes are drawn on the images shown on the right side. Below the images, there are given the accuracy values of the HPE1 approach, hand detection and HPE2 approach. [1]



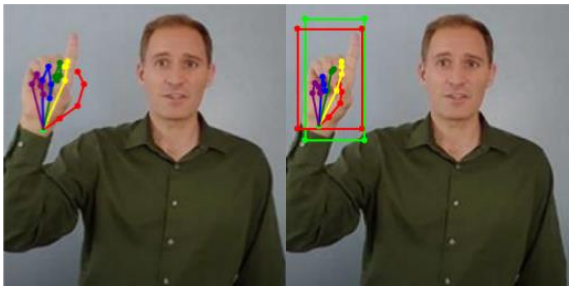
$A_{HPE1} = 90,48\%$ $A_{HD} = 79,94\%$ $A_{HPE2} = 100\%$

[1] Photo material from www.lifeprint.com. Thanks to Dr. Bill Vicars!

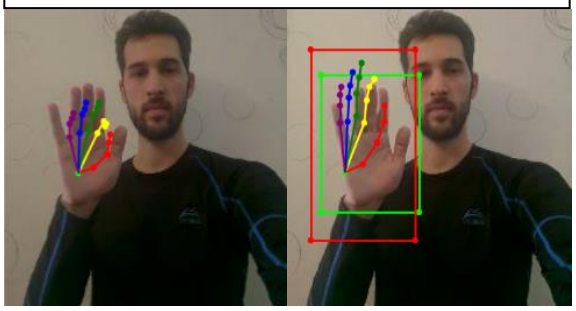


$A_{HPE1} = 4,76\%$ $A_{HD} = 79,67\%$ $A_{HPE2} = 71,43\%$

Fig. 11. Results of direct (left) and standard (right) hand pose estimation in images of Dataset 2. True (green) and predicted (red) bounding boxes are drawn on the images shown on the right side. Below the images, there are given the accuracy values of the HPE1 approach, hand detection and HPE2 approach.[2]



$A_{HPE1} = 52,38\%$ $A_{HD} = 71,65\%$ $A_{HPE2} = 57,14\%$



$A_{HPE1} = 38,1\%$ $A_{HD} = 63,6\%$ $A_{HPE2} = 85,71\%$

Fig. 12. Results of direct (left) and standard (right) hand pose estimation in images of Dataset 3. True (green) and predicted (red) bounding boxes are drawn on the images shown on the right side. Below the images, there are given the accuracy valuess of the HPE1 approach, hand detection and HPE2 approach.[3]

3.4 Discussion

The experiment clearly demonstrated the features of the standard and direct approaches. First of all, it should be noted that the keypoint detectors for the HPE1 and HPE2 systems were trained with different accuracy using datasets composed in different ways. The detection of

[2] First photo from www.lifeprint.com. Thanks to Dr. Bill Vicars!

[3] Upper photo from YouTube channel 'Chris Gorges'. Thanks to Mr Chris Gorges!

Thanks to Mr Beka Baratashvili for the bottom photo!

keypoints in the original images was studied with high accuracy when the dataset consisted of the images similar in content showing a hand in different positions. Detection in cropped images was better studied when the dataset consisted of multiple images of each hand pose, performed by different people. When the dataset was filled with many diverse images, the HPE2 approach gained an advantage, although its accuracy was not high.

Detection in the cropped images was performed more accurately than in the original images in the case of all datasets. In one case (using Dataset 1), HPE1 was more accurate. In terms of hand pose estimation speed, as expected, the HPE1 approach gained an advantage.

The direct hand pose estimation approach can be used in cases where a high-speed hand pose estimation is required in images of a certain category. HPE1 approach can be successfully used in the tasks requiring hand pose estimation in images of one or more people, or images of a certain category, etc.

Tab. 3 lists the advantages and disadvantages of both approaches.

Tab. 3. Advantages and disadvantages of HPE1 approach.

Advantages	(i) The possibility to achieve high accuracy of hand pose estimation in photographs with specific content. (ii) Relatively fast hand pose estimation. (iii) The simplicity of implementation. The approach requires solving only one problem - estimating a position of a hand in the original images.
Disadvantages	(i) Quite difficult to achieve generalization.

Tab. 4. Advantages and disadvantages of HPE2 approach.

Advantages	(i) Ability to achieve a high degree of generalization.
Disadvantages	(i) Relatively slow hand pose estimation. (ii) The complexity of the Implementation. This approach requires solving two problems - detecting a hand in an original image and estimating a hand pose in a cropped image of a hand.

4. Conclusion

The paper presents a comparative analysis of two approaches to hand pose estimation in images with specific content. The study is based on an example of a specific task that requires hand pose estimation in images showing a person speaking sign language near a camera. The goal of the study was to test the assumption that the estimation of the hand pose for a given or a similar problem could be performed faster and with almost the same accuracy compared to the standard approach if the direct approach was used. The direct approach implies estimating a hand pose directly in an original image, as opposed to the standard approach, which first finds the part of an image where a hand is shown, crops it, and then estimates a hand pose in the cropped image. This hypothesis has been tested experimentally. Both standard and direct hand pose estimation systems were constructed and compared in terms of accuracy and speed. The deep learning model, Mobilenet was trained to create the detectors. For training, three datasets of different content were created. The purpose of creating the three datasets was to examine how the hand pose estimation accuracy depends on the content of the dataset.

Efficiency coefficient was used to compare the approaches. It is calculated using accuracy and speed. A high value of the efficiency coefficient means that both accuracy and speed are high.

As a result, several conclusions can be drawn: As expected, high accuracy was achieved in the case of direct hand pose estimation using a dataset that contained samples that were similar in content but showed different hand positions. When estimating a hand pose using the standard approach, the accuracy is higher when multiple images of each hand pose are given in the dataset. In photographs with a variety of content, the HPE2 approach is obviously preferred. The assumption regarding the hand pose estimation speed was also justified. For all efficient models, the direct approach was faster. As a result, it can be stated that the standard approach is clearly preferred to achieve maximum accuracy of hand pose estimation in photographs with highly diverse content. In the case of tasks that require real-time hand pose estimation in photographs of certain content, the direct approach may be more efficient due to its high accuracy and short execution time.

Acknowledgments

This work was supported by Shota Rustaveli National Science Foundation (SRNSF) [PHDF—18-342, Optimized communication system for sign language speakers].

References

[1] M. C. Hsieh and J. J. Lee, "Preliminary Study of VR and AR Applications in Medical and Healthcare Education," *J. Nurs. Heal. Stud.*, vol. 03, no. 01, p. 1, Feb. 2018, doi: 10.21767/2574-2825.100030.

[2] Y. Zhang and O. Meruvia-Pastor, "Operating virtual panels with hand gestures in immersive VR games: Experiences with the leap motion controller," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017, vol. 10324 LNCS, pp. 299–308, doi: 10.1007/978-3-319-60922-5_24.

[3] M. Abavisani, H. R. V. Joze, and V. M. Patel, "Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training," in *Proceedings of the IEEE Computer Society Conference*

on Computer Vision and Pattern Recognition, 2019, vol. 2019-June, pp. 1165–1174, doi: 10.1109/CVPR.2019.00126.

[4] G. Garcia-Hernando, S. Yuan, S. Baek, and T. K. Kim, "First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 409–419, doi: 10.1109/CVPR.2018.00050.

[5] R. Y. Wang, S. Paris, and J. Popovic, "6D hands: Markerless hand tracking for computer aided design," in *UIST'11 - Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, 2011, pp. 549–557, doi: 10.1145/2047196.2047269.

[6] I. Papastratis, K. Dimitropoulos, D. Konstantinidis, and P. Daras, "Continuous Sign Language Recognition through Cross-Modal Alignment of Video and Text Embeddings in a Joint-Latent Space," *IEEE Access*, vol. 8, pp. 91170–91180, 2020, doi: 10.1109/ACCESS.2020.2993650.

[7] T. Chatzis, A. Stergioulas, D. Konstantinidis, K. Dimitropoulos, and P. Daras, "A comprehensive study on deep learning-based 3d hand pose estimation methods," *Appl. Sci.*, vol. 10, no. 19, pp. 1–27, Oct. 2020, doi: 10.3390/app10196850.

[8] W. Chen *et al.*, "A Survey on Hand Pose Estimation with Wearable," *Sensors*, vol. 20, no. 1704, 2020.

[9] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017, [Online]. Available: http://arxiv.org/abs/1704.04861.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

[11] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 1302–1310, doi: 10.1109/CVPR.2017.143.

[12] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1653–1660, doi: 10.1109/CVPR.2014.214.

[13] G. CHAGANAVA and D. KAKULİA, "Keypoint Detector Retraining Techniques for the Communication System of Sign Language Speakers," *Eskişehir Tech. Univ. J. Sci. Technol. A - Appl. Sci. Eng.*, vol. 21, pp. 74–86, 2020, doi: 10.18038/estubtda.822295.

[14] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. Da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electron.*, vol. 10, no. 3, pp. 1–28, 2021, doi: 10.3390/electronics10030279.

[15] "Home | TensorFlow Hub." https://tfhub.dev/ (accessed Aug. 24, 2021).

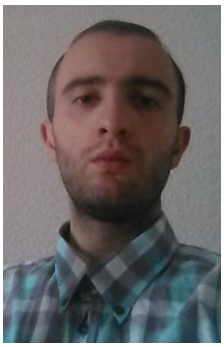
[16] Indiana University, "EgoHands: A Dataset for Hands in Complex Egocentric Interactions | IU Computer Vision Lab," 2020. http://vision.soic.indiana.edu/projects/egohands/ (accessed Jul. 30, 2021).

[17] "GitHub - victordibia/handtracking: Building a Real-time Hand-Detector using Neural Networks (SSD) on Tensorflow." https://github.com/victordibia/handtracking (accessed Jul. 30, 2021).

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[19] "cansik/yolo-hand-detection: A pre-trained YOLO based hand detection network." https://github.com/cansik/yolo-hand-detection (accessed Jul. 30, 2021).

[20] C. Y. Wang, H. Y. M. Liao, P. Y. Chen, and J. W. Hsieh, "Enriching variety of layer-wise learning information by gradient combination," in *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, 2019, vol. 2, pp. 2477–2484, doi: 10.1109/ICCVW.2019.00303.



Guram Chaganava received the B.S. and M.S. degrees, from Tbilisi State Univ. in 2015 and 2017, respectively. His research interest includes embedded systems, image processing, deep learning.