Finding key points in a hand image using heatmaps

Finding key points in a hand image using heatmaps

D V Arslanova[1], A E Grishin[1], A I Denisov[1], D Z Galimullin[1], N V Denisova[2], R A Makarov[3] and M E Sibgatullin[4, 5]

[1]Institute of Computational Mathematics and Information Technology, Kazan Federal University, Kazan, Russian Federation
[2]Kazan State Power Engineering University, Kazan, Russian Federation
[3]Kazan Innovative University named after V.G. Timiryasov (KIU), Kazan, Russian Federation
[4]Tatarstan Academy of Sciences, Institute of Applied Research, Kazan, Russian Federation
[5]Institute of Physics, Kazan Federal University, Kazan, Russian Federation

E-mail: andrey.g.1991@gmail.com

Abstract. This paper presents an approach to finding key points in the image of a hand using heat maps. For this, a convolutional model of a neural network and a training method on heatmaps were combined. An open bank of palm photos, supplemented by a set of own images and synthetic data, was used as a dataset. Direct and inverse transformations of two-dimensional coordinates on the plane into heat maps with a 2D Gaussian function centered at coordinate points were applied, data was prepared, the model was trained and tested. As a result of this approach, a neural network model able to recognise real world images was obtained.

1. Introduction

Detection of a hand in space and finding key points on it is one of the most promising tasks of computer vision [1-3]. Developments in this direction can be used in the design of human-machine interaction, the creation of virtual reality applications, recognition of the sign language, *etc* [4-7].

This paper describes a method for detecting key points on the palm plane using “heat” maps. This will allow us to proceed to determine the orientation of the hand relative to the plane of the smartphone, from the camera of which a video stream is broadcast in real time, which, in turn, can be successfully used in VR technologies.

2. Articles and turnkey solutions

The development of Google [8] can be called one of the most successful in this area. It implements several technologies that work together: palm recognition, identification of key points and classification of gestures. MediaPipe framework designed by Google successfully works on a smartphone and operates in real-time operation of the video stream.

To train the model to find 21 key points, the researchers manually marked about 30 thousand images. Nevertheless, this development can be called exclusively commercial and not bearing scientific research value, in view of the fact that it is a closed source product, without the ability to study or make changes to it. Another significant drawback that impacts the use of MediaPipe in applications is its incorrect work in imperfect conditions: when the hand does not completely fall into the frame, the application either concentrates all the points in one place (Figure 1), or stops detecting them at all in the image.

Alchera's development has similar functionality and is based on the article [9]. It describes the use of Convolutional Pose Machines [10], which finds many key points of one hand from different types. Then, using triangulation, a three-dimensional location of each point is obtained, which ensures high accuracy when transferred to a plane. The main drawback of this technology is in huge requirements for data collection: in the original work, a system of 31 HD cameras was used to obtain images? so modifications or reproduction of this approach pose a great challenge.

Instead, we concentrated on a more simple approach. A network training method similar to that described in this article was proposed in [11], where heat maps were used to find key points on the face and further build facial contours.

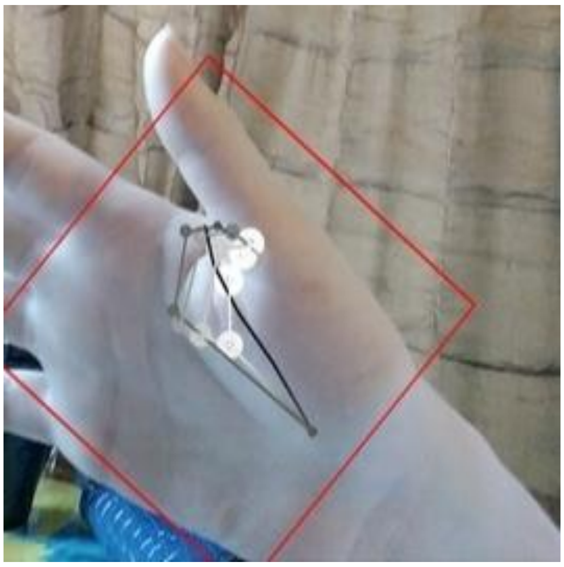
Figure 1. Broken MediaPipe Case Example

3. Data collection
There are several datasets with hand images. Most of them are designed to create classifiers that define gestures. For approach with heatmaps detection, images with the unfolded plane of the hand were needed, which makes it possible to train the model for recognizing key points. The basis was an open dataset with images of hands "11k Hands" [11, 12]. Examples from this dataset presents at Figure 2.

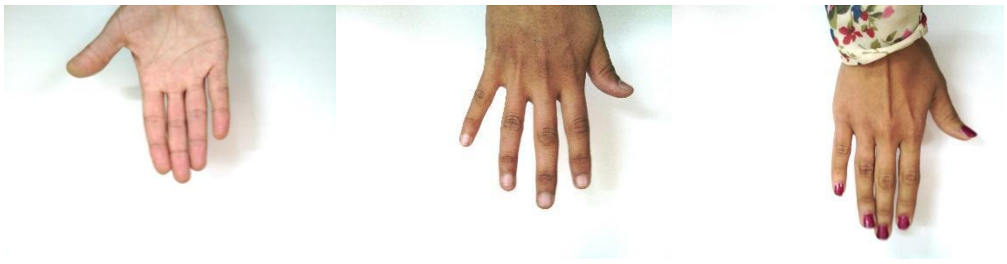
Figure 2. Examples of images from the dataset "11K Hands"

For best results, some data was excluded, and only images with the back of the hand were saved. We also assembled our own dataset (Figure 3) containing 1000 photographs: a device (Figure 4) was designed, one side of which is at an angle of 30 degrees, and the other - 45. With it, photographs were taken of hands with different angles of inclination and rotation.

The dataset was supplemented with images artificially obtained by rendering a 3D model of the brush (Figure 5). To obtain a large amount of data, several 3D models were created and a script was written that allows to perform batch rendering with a given angle of inclination and rotation.

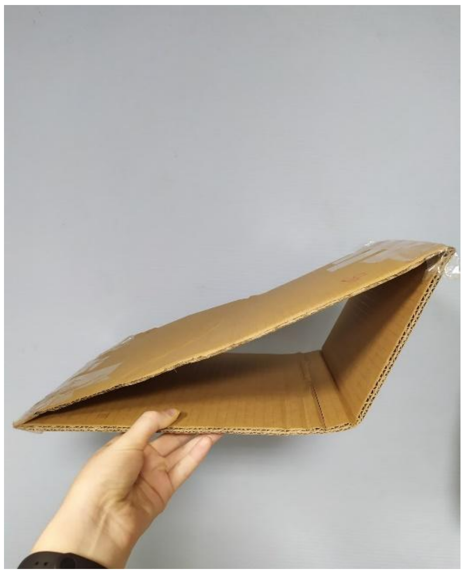
Figure 3. Device used for data collection

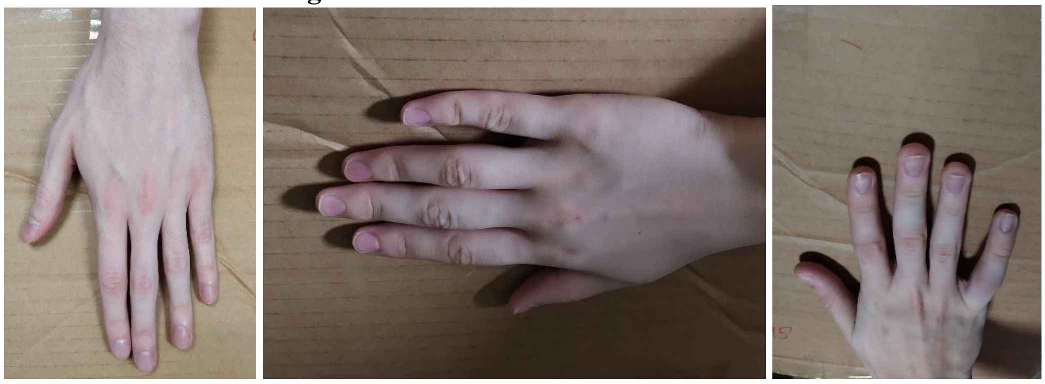
Figure 4. Received images

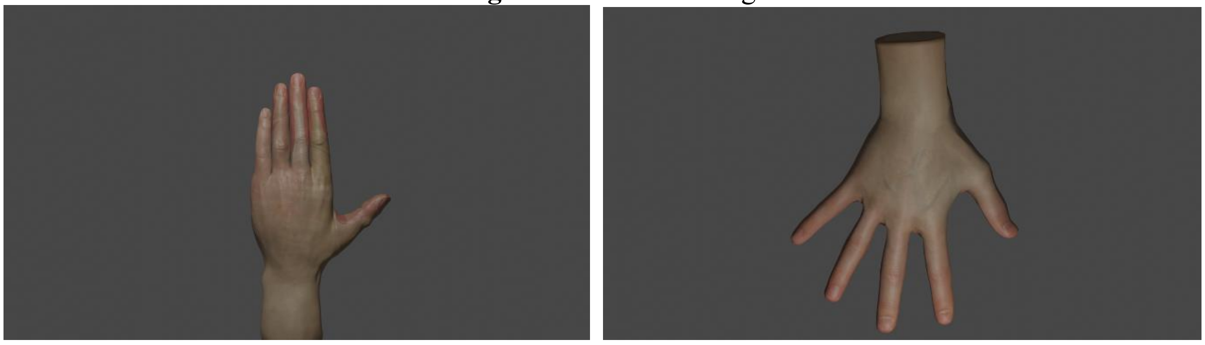
Figure 5. 3D models used to obtain synthetic data

4. Data markup

Images from the "11k Hands" dataset were pre-marked using the "auto-tagger" for hand images, which is a model of a network trained in Caffe. Due to the imperfection of this markup tool, some of the original dataset could not be qualitatively marked, after sifting out low-quality cases, 7 thousand marked images were received.

Assembled images were also marked out manually using the free Labelme labeling tool developed in the Python programming language (Figure 6).

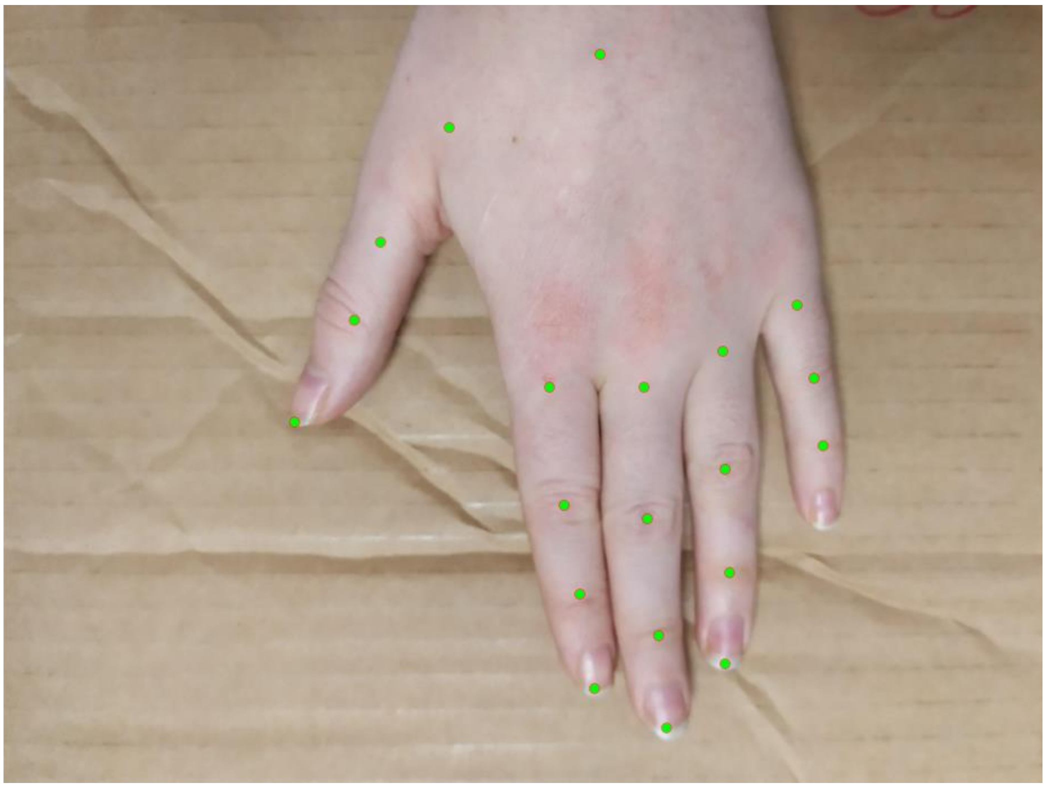

Figure 6. Hand markup visualization in the Labelme interface

5. Training data preparation

In order to prepare the data for training, certain manipulations were carried out with them.

For some of the photos, the background was replaced using the implementation of OpenCV tools in Python to achieve greater network stability (Figure 7). The background replacement algorithm can be described as follows:

1. Edge detection: Creating the basis for a future mask by working with a gradient calculation operator on the original image using the Sobel function.
2. Fill: Fills the outer area with black.
3. Noise removal: unadorned islands of pixels are removed, borders are smoothed.
4. Final stage: The mask is binarized, blurred a bit and the output alpha channel is obtained.

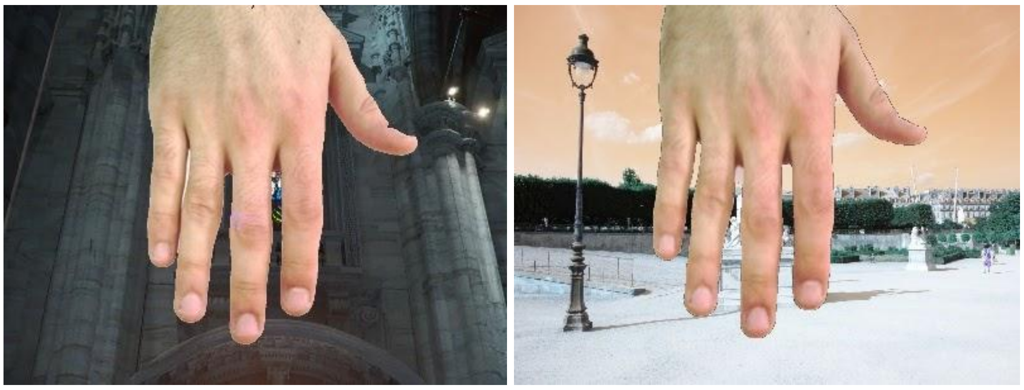

Figure 7. Examples of images of hands with a changed background

Using the color segmentation method, the hand images themselves were directly cut from the original photos, with the corresponding transfer of the coordinates of the key points to them, thanks to which we managed to get rid of excess image fragments (Figure 8).

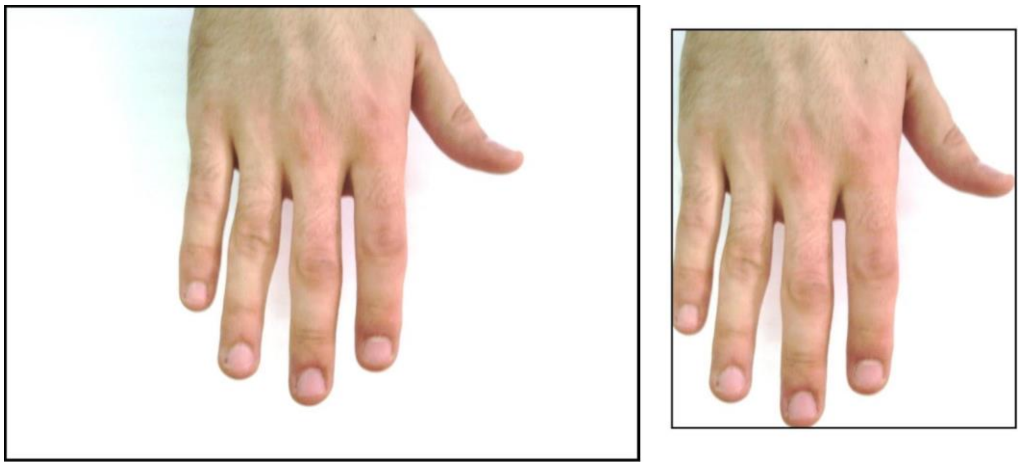

Figure 8. Image before and after cutting out the main part

The cut out images were compressed to a size of 96x96, also with the corresponding correction of the coordinates of the key points (Figure 9).

Figure 9. Compressed image

Next, the image was taken in a single-channel color format, and heat maps were constructed for all the coordinates of the key points, each of which is a separate image with a resolution of 96x96, with a black background, with a Gaussian printed on it with the center in the coordinate value of the corresponding key point (Figure 10). Thus, for twenty key points, it was necessary to create twenty heat maps.

For example, the value of a point with coordinates ***(x, y)*** of a heat map for a key point with coordinates ***(x_0, y_0)*** will be calculated according to the expression (1):

$$\frac{1}{2\pi\sigma^2} expexp\left(-\frac{\left(x-x_0\right)^2+\left(y-y_0\right)^2}{2\sigma^2}\right). \tag{1}$$

Where the parameter is chosen arbitrarily based on the context of the task. However, it is important to understand that if the value is too low, the heat map will turn out to be too sparse, and if too high, the model will no longer concentrate on the coordinates of the key point. In our case, the value was taken equal to 5, the choice was due to the fact that the width of the finger is approximately 10 pixels, so that a Gaussian with a value of 5 when applied will cover the entire width of the finger.

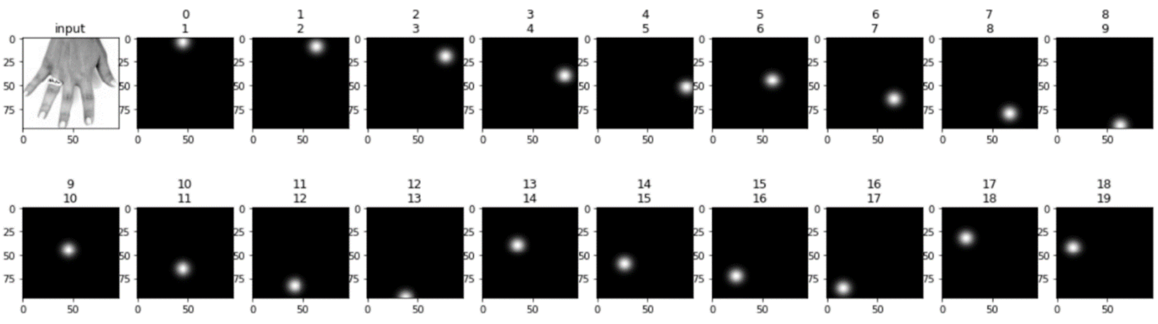

Figure 10. Heatmaps of the keypoints

It is very important to supplement existing training images in order to improve the model's performance on future validation data. Images and their heatmaps were augmented to increase the size of the dataset and to add variety to the data, in particular horizontal reflections and affine transformations (Figure 11).

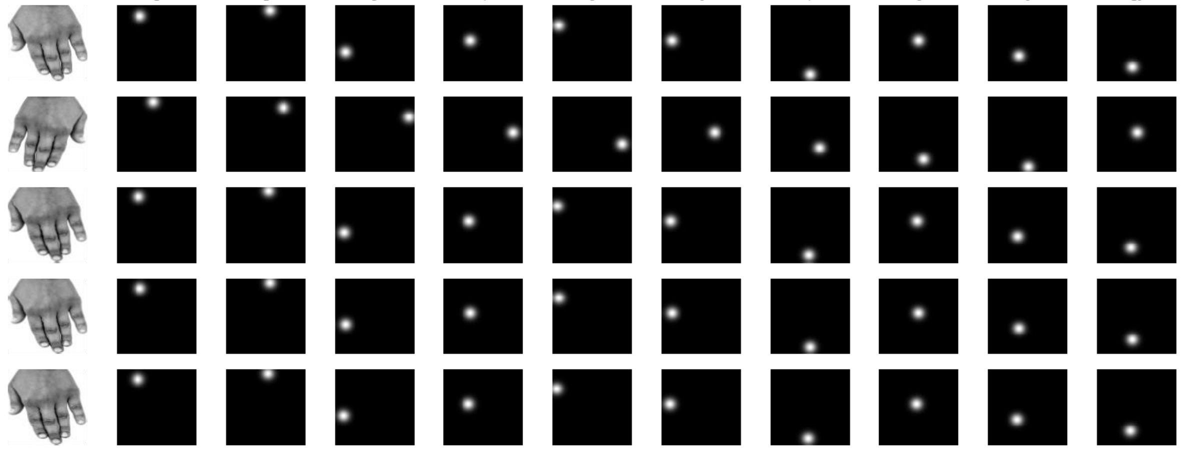

Figure 11. Data augmentation

6. Network Model Training

By the time the network began training, there were 2,551 labeled images available for training and 283 for testing. The network model is shown in Figure 12.

An image in 96x96 resolution in single-channel format is fed to the input of the network model. And at the output of the network we get 20 images of heat maps. To obtain the coordinates in the format ***(x, y)***, the image data is converted into coordinates by finding the center of the Gaussian on the heat map.

The training was carried out on the open service Google Colab on the GPU (NVidia Tesla K80). This model was trained in three hundred eras with a batches size of 32, with a separation into a training and validation sample in the ratio of 0.9. As a result of training, we obtained the value of the loss function on the training sample equal to 0.0279 and on the validation sample equal to 0.0726 (Figure 13). Network training took 4 hours.

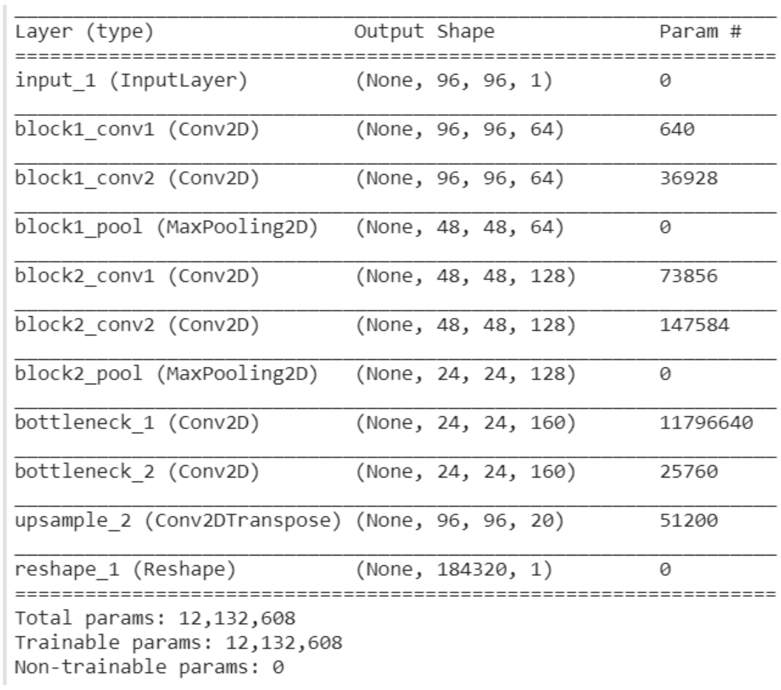
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 96, 96, 1)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 96, 96, 64)        640
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 96, 96, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 48, 48, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 48, 48, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 48, 48, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 24, 24, 128)       0
_________________________________________________________________
bottleneck_1 (Conv2D)        (None, 24, 24, 160)       11796640
_________________________________________________________________
bottleneck_2 (Conv2D)        (None, 24, 24, 160)       25760
_________________________________________________________________
upsample_2 (Conv2DTranspose) (None, 96, 96, 20)        51200
_________________________________________________________________
reshape_1 (Reshape)          (None, 184320, 1)         0
=================================================================
Total params: 12,132,608
Trainable params: 12,132,608
Non-trainable params: 0
```

Figure 12. The structure of the neural network model

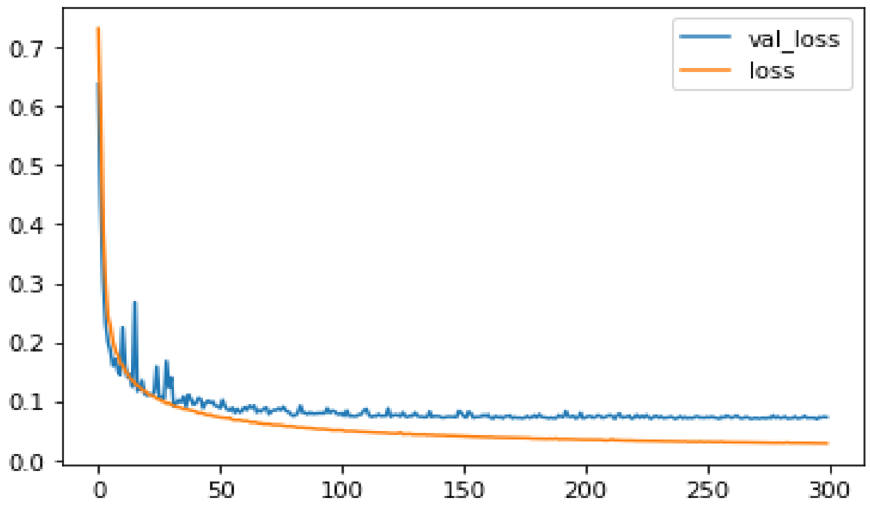

Figure 13. Loss function graph

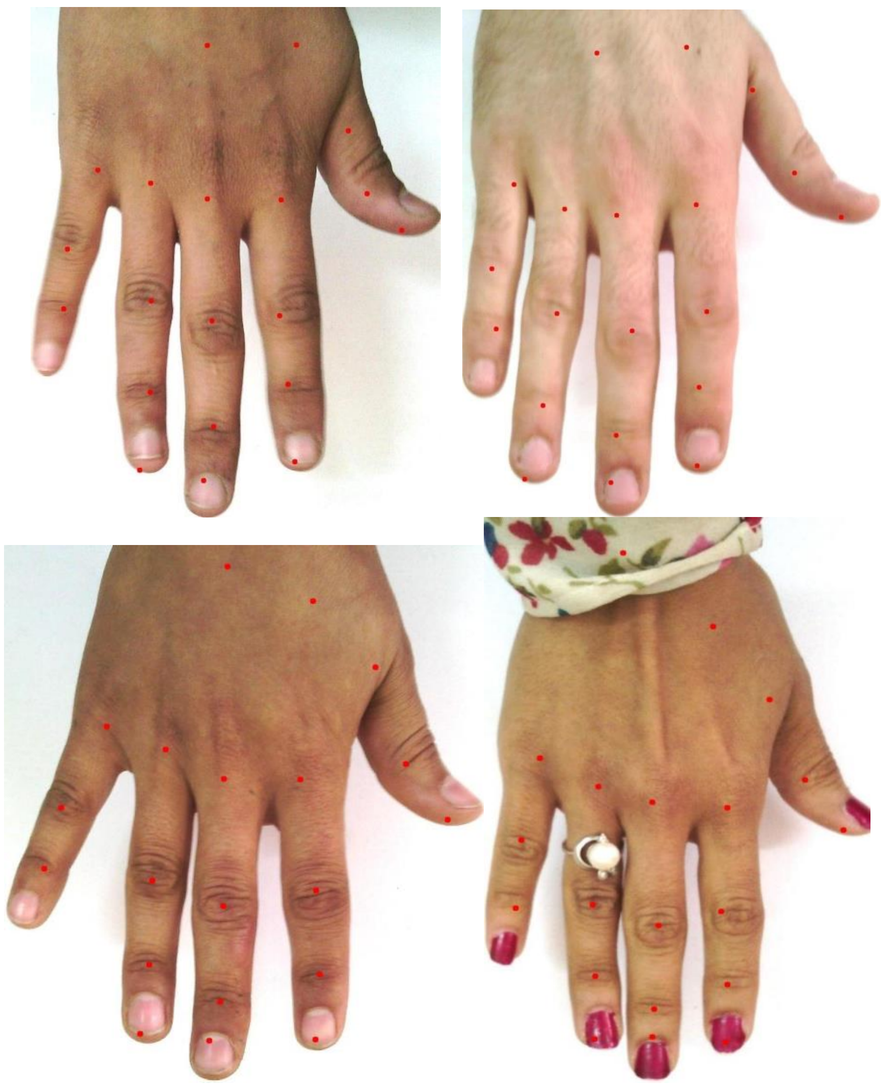

Figure 14. Network operation result

7. Network Model Testing

A test environment has been developed to test the health of the network model. If the distance between the predicted point and the known true value of the point differs by more than 9 pixels (the average half of the finger thickness in the image resolution is 96x96), then we assume that the point is incorrect. If at least one point is incorrect in the image, then we assume that the image is recognized incorrectly. As a result of this check, it was found that 40% of the 120 test images are recognized correctly, and among unsuccessful 31% of the total, the number is recognized with an error of 1-2 points. An example of the network is shown in Figure 14.

8. Conclusion

As a result of the work done, we got a trained neural network model that is able to recognize key points in the image of a hand. In the future, it is planned to improve this model to increase stability, as well as to train new ones for new capabilities, in particular for directly finding the hand in the photograph, as well as for finding the angle of the arm in space.

References

[1] Tompson J, et al 2015 *Proc. Conference on Computer Vision and Pattern Recognition* (10.1109/CVPR.2015.7298664) pp 648-656

[2] Toshev A and Szegedy C 2013 *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 10.1109/CVPR.2014.214.
[3] Xiao B, et al 2018 *Simple Baselines for Human Pose Estimation and Tracking* 10.1007/978-3-030-01231-1_29.
[4] Newell A, et al 2016 *Proc. Computer Vision – ECCV 2016: 14th European Conference* (Amsterdam, The Netherlands) pp 483-499
[5] Belhumeur P, et al 2013 Localizing parts of faces using a consensus of exemplars. TPAMI.
[6] Lu C and Tang X 2015 *Proc. of the TwentyNinth AAAI Conference on Artificial Intelligence* (AAAI Press) pp 3811–3819
[7] Zhou E, et al 2013 *Proc.IEEE International Conference on Computer Vision Workshops* pp 386–391
[8] Bazarevsky V and Zhang F 2019 *On-Device, Real-Time Hand Tracking with MediaPipe* (Google Research, https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html)
[9] Simon, et al 2017 *Proc. Conference on Computer Vision and Pattern Recognition* (10.1109/CVPR.2017.494) pp 4645-4653
[10] Wei, et al 2016 *Convolutional Pose Machines*
[11] Bulat A and Tzimiropoulos G 2017 *How Far are We from Solving the 2D & 3D Face Alignment Problem? (and a Dataset of 230,000 3D Facial Landmarks)* 10.1109/ICCV.2017.116
[12] Bulat A and Tzimiropoulos G 2016 *Proc. ECCV 2016: 14th European Conference (Amsterdam, The Netherlands)*