Project Part 3: Hand Gesture Recognition in Real Time

Isac Nordin, Kelly Ma, Christian Rutgersson
Chalmers University of Technology
(Dated: May 27, 2022)

In modern societies, interactions between humans and computer systems are an integral part of infrastructure. Thus, novel methods of information transfer may bring with it great improvements in many areas of society. In this project a hand gesture recognition algorithm has been developed and tested. The algorithm is split into two parts: classic image segmentation, and a convolutional neural network that classifies the segmented image. Three different methods were used to find the hand in the image: a box in which the user places his/her hand, a skin color based segmentation, and a pre-trained neural network acquired from the MediaPipe library. The classification performance was tested by three different individuals at separate locations. The results showed that the classification accuracy varied based on hand gesture, and that the static box, skin color segmentation, and MediaPipe hand detection methods ended up with average accuracies of 71 %, 58 %, and 53 % correct classifications respectively. The results were not determined to be good enough for the program to be used in applications such as a sign language translator. However, it would perform well in simpler tasks involving the user answering yes or no to questions, for example.

I. INTRODUCTION

For many years, mechanical inputs such as keyboard and mouse were the only viable options for human-computer interaction (HCI). However, new possibilities are emerging followed by the rapid growth of both hardware and software [1]. Visual methods of HCI are emerging in many applications. For example, eye recognition is now common in smartphones. These methods aren't just a gimmick, as they come with some unique advantages. One is that some disabled people might find easier access to the technology used in the modern world. Another advantage is that using gestures instead of touch could reduce spread of bacteria and viruses in many public spaces [2]. This is something that is being realized by some airports. For example, visual based methods are beginning to get implemented in check-in kiosks [3].

There is much value in trying to widen the array of possible applications for this type of technology. The central factor here is price. Therefore, this project sets out to create a good hand gesture recognition algorithm to be used for a simple laptop and web camera in real time. In the following section (II), the methods used to achieve this algorithm and the tests used to evaluate it are described. In the section III, the results from the tests are presented. Afterwards, in sections IV and V, the results, further work, and conclusions are discussed.

II. METHODS

The development of a real-time hand gestures program is split into two parts. First, segmentation of pixels such that the hand is extracted from the frame captured with a web camera. Secondly, classification of the segmented images into ten pre-defined hand gestures with a convolutional Neural Network (CNN).

A. Background Subtraction

Background subtraction might be of interest in many different scenarios and is essentially used when the background is distracting. The base of it is to subtract the reference frame from the current frame and thereby detect any newly added object (the foreground), after binary thresholding. For this specific task, the reference frame f_r was chosen to be a weighted average of a frames sequence $f_i, i = 1, 2....n$, as shown in equation 1, with weight $\alpha = 0.5$ and the numbers of frames $n = 50$.

$$f_r^i \leftarrow (1-\alpha) f_r^{i-1} + \alpha f_i \tag{1}$$

The frames sequence gets collected automatically during the first instance of running the program, but it is also possible to reset and recalculate the background at a later time with a keystroke.

The quality of background subtraction was tested separately by varying backgrounds and seeing how clean the segmentation became, both for almost uniform backgrounds and those with many different variation in color.

B. Hand Detection

Background subtraction can efficiently extract objects of interest when they are the only item added to the reference frame. However, in this case, the hand is not the only non-background part. Once the user moves into frame, the upper body, arm, and face becomes visible together with the hand. A hand detection algorithm to isolate the hand is thus needed in order to get a resulting segmentation that doesn't include any irrelevant body parts.

1. *Static Box*

The most straightforward way to achieve hand detection is to place a visible static box in the frame and ensure that the user always puts their hand there and remove everything outside of the box, as shown in figure 1.

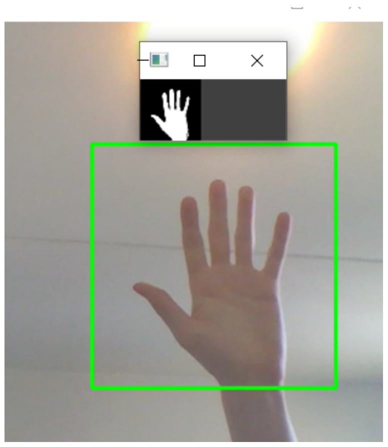

FIG. 1: A user placing their hand inside a static box In this image, the static box hand detection together with the background subtraction can be seen.

2. *Color-based Segmentation Combined with Convex Hull*

Another alternative with higher flexibility is color-based thresholding combined with convex hull. In the simplest terms, this method first detects and segments pixels based on a pre-defined skin color interval, and then finds the largest area of skin color in the image and detects this as the hand. However, a problem emerges when the user's face is captured by the camera, as it can be mistaken by the algorithm to be the hand. To solve this, Haar cascade was used to remove the face from the frame.

Haar cascade essentially works by sliding a sequence of filters (e.g. figure 2 a) across the image. The mean of the pixel values of the white and black boxes was computed for the first filter in the sequence. If the means closely match a pair of pre-defined numbers (typically one and zero), more filters are sequentially applied in the same place. Otherwise, the detection window moves further to the next region, as shown in figure 2 b). If all filters return the correct output, a face is detected and removed from the image. [4]

The thresholding was done in HSV (Hue, Saturation, and Value) and YCrCb (luminance, and red & blue chrominance) color spaces according to the criteria shown in equation 2. The criteria were originally found from different sources online for skin detection, and modified by testing different values [5].

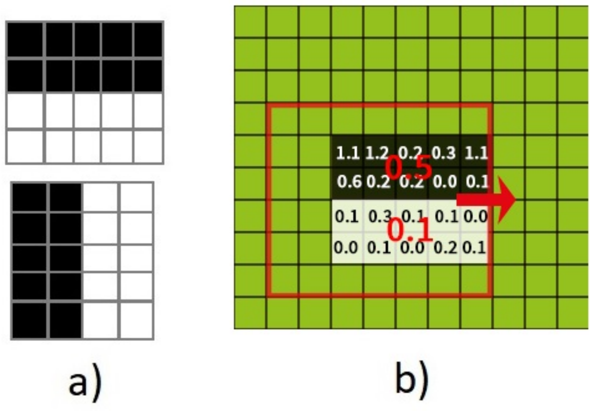

FIG. 2: a) shows 2 simple and possible kernels. b) The kernel slides across the screen and checks if the mean pixel value inside the white&black part should be close to 1&0 and if so it has detected a face.

$$\begin{aligned} 0 < H < 17, \quad 15 < S < 170, \quad 0 < V < 255 \\ 0 < Y < 255, \quad 130 < Cr < 85, \quad 85 < Cb < 135 \end{aligned} \tag{2}$$

To reduce noise and therefore improve the skin color based segmentation, the image was first of all pre-processed with a gaussian filter. After the application of color-segmentation, the image was also passed through an opening filter (morphological operation) with a square structuring element of size 3-by-3.

Provided the thresholding succeeds, the hand should be the only part that remains in the frame. The final step of this type of hand detection was to apply a convex hull, such that the largest object in the image with skin color (the hand and arm) was encased in a polygon as in figure 3, and passed to a CNN model to predict the gesture. The convex hull tended to falsely identify the arm as a part of the hand, so for optimal performance, it is highly recommended to wear longsleaved shirts.

The resulting image is somewhat dependent on ambient light, which is why it was only used for hand detection, and not background subtraction and hand segmentation. Multiple tests were therefore made to investigate how different background lighting settings affected it. The test was conducted by primarily moving a lamp in different directions but a comparison was also made with natural lighting.

3. *MediaPipe*

Media pipe is a library that handles different computer vision applications with use of their own pre-trained neural networks (more information available on their website

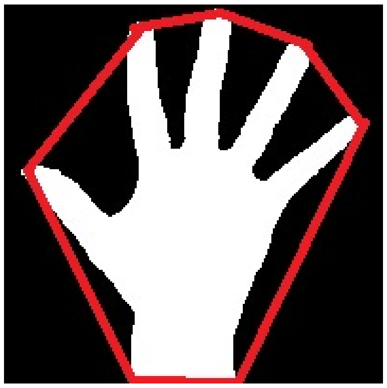

FIG. 3: A segmented hand with a red Convex hull drawn.

[6]). One such application is to locate the different fingertips and joints when a hand is on screen. This is used to locate the minimum and maximum x & y position of different hand joints to create a box surrounding the whole hand.

C. Dataset

The dataset used during training was taken from Kaggle, and contains 18000 images of 20 different gestures, 900 images per gesture [7]. However, only ten gestures were used in this project due to them being easy to replicate, labeled as seen in figure 4. Note that all images in the dataset were made by the same person, were centered, had a fixed rotation & direction towards the camera, and contained no noise. The difference between images with the same label were quite small. To make the dataset more varied, and applicable for real world usage, all images were randomly rotated with $\theta \in$ [-40,40] degree and translated in a random direction with $x \in$ [0,10] pixels. The data format was a binary image of size 50x50 with pixel values being either zero or one.

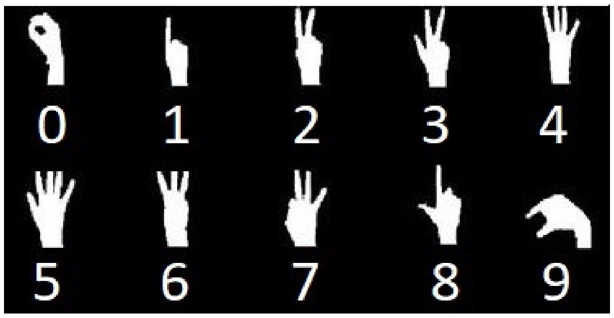

FIG. 4: Examples of gestures from the training dataset.

D. Convolutional Neural Network

In order to classify the hand gesture images, the segmented frames were passed through a convolutional neural network (CNN) trained on the dataset described in section II C. Early stopping was implemented with the intention to prevent overfitting. Overfitting is a problem where the accuracy of the network on unseen data is lowered due to it learning to classify based on redundant features in the training data.

Before the segmented hand image was passed through the CNN model, it was resized and normalized to have the same format as those in the training dataset. The model returns an output vector with a length equal to the number of classes, and the index of the maximal value in the output vector identifies the predicted gesture which was displayed on the screen.

The architecture used is a standard CNN model as seen in figure 5, where features of the image such as edges and corners were extracted by convolutional layers and later classified by multiple dense layers. The specific way a convolutional layer works is by applying a convolution with a trainable weight filter as seen in figure 6. For each convolutional layer many filters are applied to the same input to gain many different features. After a convolutional layer maxpooling and dropouut can be used. Maxpooling reduces the feature images' size by generalizing, seen in figure 6 which reduces trainable parameters and training time. Dropout sets a fraction of weights to zero which reduces overfitting.

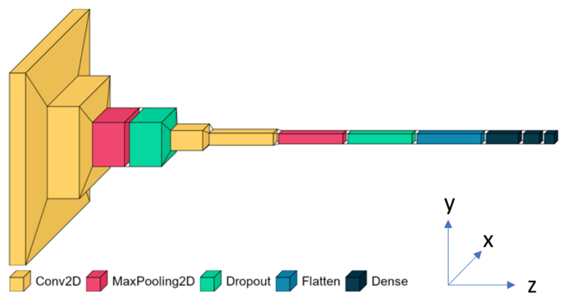

FIG. 5: Convolutional neural network (CNN) architecture to classify hand gestures. On the far left is the input image of size 50x50x1. At each layer convolutions with trainable filters takes place which changes the dimension $(X, Y, Z) \to (\frac{X}{2}, \frac{Y}{2}, 2Z)$. After a few convolutional layers the input is flattened to be made into a dense (fully connected) layer. The figure represents the scaling of the layers correctly, including the depth (number of channels) along the z-axis.

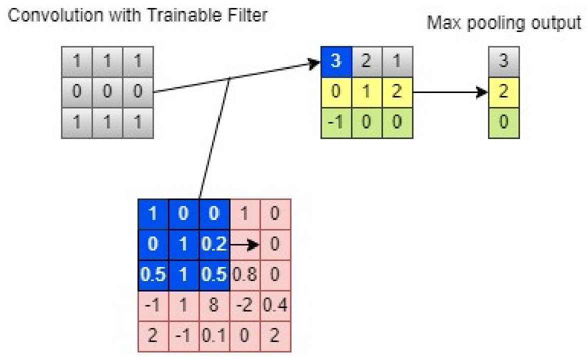

FIG. 6: Convolutional layer and maxpooling showcased. A convolution with a trainable filter is made getting a new image or feature. A maxpooling layer then generalizes the image by picking the maximum value of e.g. a row.

E. Frames per second

A test was made to see how many frames per second (FPS) the program could capture while three different hand detection algorithms were used, with and without the CNN gesture prediction. The FPS was only tested on a single laptop, with no GPU (graphical processing unit) accelerator enabled, which would speed up certain parts such as CNN gesture prediction. The results were expected to depend on the computer's performance, but only the comparison between different methods was of interest.

F. General Accuracy test

The gesture recognition program was tested by checking the prediction accuracy of gestures under ideal lighting, by three people at separate locations, and the results were averaged. The CNN used in the program had already been tested using the dataset's validation set. However, it was still necessary to test the performance of the whole program. As in a real application, the input to the CNN model could differ from those in the dataset, which affects the prediction performance.

One test was made for each hand detection method by holding a gesture directed towards the screen, and then rotating the hand back and forth 45 degrees around the vertical axis, and thereafter around the axis perpendicular to the palm of the hand. The accuracy was defined as the fraction of video frames during which the CNN classifies the current gesture correctly.

G. Binary Accuracy Test

In order to compare the results of the program on a simpler task than classifying ten gestures, a smaller and less rigorous test with only two hand gestures (gesture zero and one in figure 4) was also conducted. The same CNN model as the one used in the previous test was used. But instead of taking the index of the maximal value in the output vector, only two elements from the vector were considered and compared with each other. These two elements corresponded to the chosen hand gestures.

III. RESULTS

The background subtraction method worked differently depending on how uniform the background was. If the background was a plain white wall, the subtraction worked almost perfectly with no noise. However, if the background included sharp edges and corners such as the edges of a chair, the background subtraction would be satisfactory, but worse than clean background. This can be seen in figure 7 which showcases two backgrounds, The more contrast in the background, the worse the background subtraction.

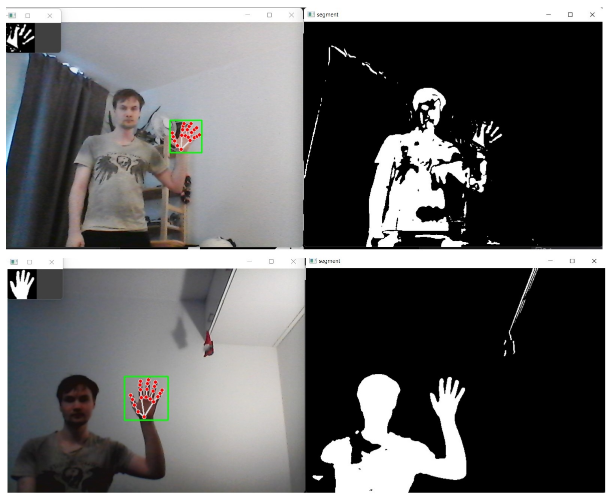

FIG. 7: Background subtraction for a almost uniform background as well as a background with some different variation in color.

The skin color segmentation worked well consistently if a lamp was facing the camera and even better if there was sunlight lighting up the area, as seen in figure 8. As can also be seen in the figure, the method gave quite different segmentations depending on the lighting, which made it fairly unreliable as a whole.

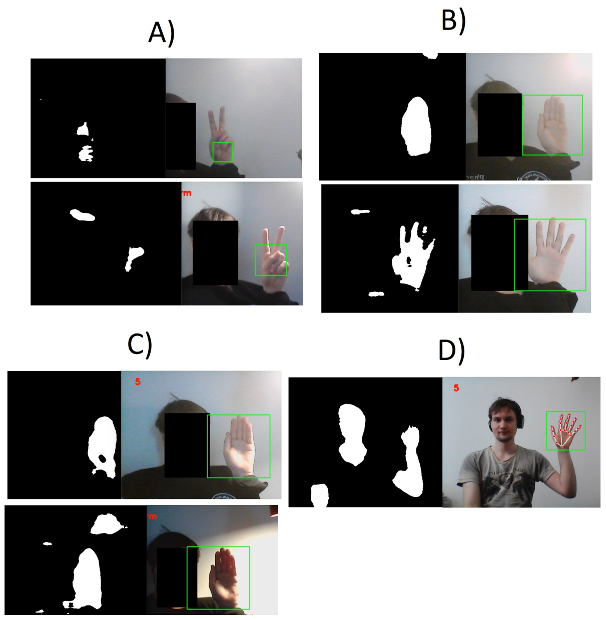

FIG. 8: How different background lightnings affected skin segmentation for hand detection. A) regular roof lamp, B) lamp directed towards camera, C) lamp directed towards hand, D) sunlight

A. Frames per Second

The video frames per second (FPS) were tested on a single laptop for the three different hand detection algorithms, both with and without the CNN gesture prediction and can be seen in table I.

TABLE I: The FPS achieved while using different configurations for the hand gesture recognition algorithm.

FPS	**Static box**	**Skin segm.**	**MediaPipe**
Without CNN	31	24	12
With CNN	13	10	7

B. Test Results

The results from the test of the program's prediction accuracy of gestures are summarized in table II. As can be seen, certain gestures were more challenging for the network to classify than others, such as gestures nine and ten. Summarizing the results for all gestures, the overall success rate for the static box method, the MediaPipe method, and skin color-based segmentation was 71%, 53%, and 58% respectively.

When using the network to only classify gesture zero and one according to figure 4 the accuracy rose dramatically to roughly 99% as can be seen in table III.

TABLE II: Hand gesture recognition accuracy for the three different methods of hand detection, and the average accuracy of all methods.

Gesture	0	1	2	3	4	5	6	7	8	9
Static box	0.74	0.69	0.76	0.62	0.87	0.78	0.69	0.51	0.49	0.94
Skin seg.	0.66	0.50	0.82	0.49	0.84	0.86	0.77	0.26	0.20	0.44
MediaPipe	0.89	0.36	0.73	0.52	0.91	0.73	0.51	0.16	0.17	0.32
Average	0.76	0.52	0.77	0.54	0.87	0.79	0.66	0.31	0.29	0.56

TABLE III: Binary classification accuracy result

Gesture	0	1
Static box	0.98	0.98
Skin seg.	0.99	0.96
MediaPipe	0.99	0.99
Average	0.99	0.98

IV. DISCUSSION

As could be seen in the result section, the maximum achieved accuracy when using all hand gestures was 71 %. For many applications this might not be good enough. It should be noted, though, in some real world applications, the user wouldn't rotate their hand as much as was done in the testing and the obtained results should be seen as a worst-case scenario. Also, very high accuracy has been seen when using two hand gestures indicating that a number of gestures in between ten and two can give acceptable results.

Comparatively, the static box method achieved the best accuracy out of the three hand detection algorithms, but at the cost of flexibility. Given that the result is based on tests from only three different individuals, it has a sizable error compared to a test with more people. Therefore, no definite conclusion should be made at this stage on whether the skin segmentation method can actually be seen as better than the MediaPipe method.

A. Background Subtraction

The background subtraction worked very well with uniform backgrounds whose color was not very similar to humans' skin, and having acceptable performance when using a detailed background with color variation. A way to improve the background subtraction would be to use morphological operations like opening to remove noise created from detailed backgrounds.

B. Hand Detection

No perfect hand detection method was found in this project, as all tested methods had their own disadvantage. The drawback of having a static box is obviously flexibility as both the placement of the hand and the distance between user and camera becomes fixed.

MediaPipe manages to predict the placement of a hand skeleton correctly most of the time and in almost all circumstances. However, the model is pre-trained with a non public training data and would be hard if possible to train it further and improve upon it.

There were some troubles however with the skin segmentation and convex hull method. First a need to use a long sleave to avoid detecting the arm is not preferred for some applications. Secondly the detection accuracy strongly depends on the background lighting. The hand detection either fail or become unstable when not having sunlight or using a lamp directed towards the camera.

The detection also fluctuated due to the camera's measurement error, which could be mitigated by averaging the last five frames. The hand segmentation often failed if the head was looking to the side. This is however acceptable since people often look forward when using gestures, especially if it is an interactive gesture.

Lastly, the skin color based hand detection algorithm was only tested on three individuals with similar skin colors and would need to be tested on more people with different skin colors to determine if the thresholding needed to be modified. A way to improve the hand detection's reliability would be to use a colored glove to easily segment the hand.

C. Hand Gesture Performance

The CNN was able to predict gestures well quite often, however there were some troubles. First being that the background subtraction didn't work perfectly, causing noise in the segmented image. It is also very likely that the hand detection methods could have made the box too big, or too small, causing differences with the training dataset. These sorts of differences between the dataset and the users' hands after segmentation could be fixed in few ways. Noise could be introduced to the training data, as well as using a self-created dataset which contains for example more variety in hand sizes and gesture rotations.

The last problem is that the network always predicts a hand gesture, even while no hand is shown in the image, which would be problematic if the gestures are used interactively.

A way to get more insight into which hand gestures the CNN thinks are similar a confusion matrix could have been used. This knowledge could assist in increasing the accuracy of the algorithm by removing one of two gestures that were easily misclassified as each other. Of course, this reduces the number of gestures available, but if increased accuracy is prioritized, this method could be useful.

The accuracy could have been increased if the hand gesture classifications from the last ten or so frames was collected and filtered to give a result. For example only displaying the final prediction if a majority of frames gave the same prediction.

D. Frames per Second

The MediaPipe method had the worst fps, probably due to its reliance on neural networks with heavy computations. On average ten frames per second were accomplished with hand gesture recognition, which is good enough for the program to be usable as it is, however, depending on the application, higher FPS might be needed. To improve the CNN prediction speed, less trainable parameters e.g. convolutional layers could be used, or a better computer with e.g GPU-acceleration to speed up calculations (although that might fall outside the scope of this project, as the algorithm should work well for a normal laptop). The last and easiest improvement to the frame rate would be to locate and predict hand gestures only every second or third frame.

E. Further Work

Further work within the subject could go in many different ways. The hand detection could be improved with traditional methods, perhaps random forest segmentation. Another possible focus is optimizing code and calculations, or using special equipment e.g IR-camera and special background. Yet another area of interest could be usage of multiple cameras to improve recognition quality by using information about depth. Also, an end-to-end neural network approach, where a CNN is trained on just the downsized frame captured by the camera, could be tested. Lastly, an area that can be worked on is to apply the hand gesture recognition, and activate functions such as zooming when doing e.g a thumbs up.

V. CONCLUSION

From the results of the project it can be concluded that hand gesture recognition can be achieved with such simple tools as a camera and a non high performance computer, which was the purpose of the project. One can also conclude that there exists a trade-off between the number of gestures used and the resulting accuracy. Further analysis would be needed to find the maximal number of gestures while preserving acceptable accuracy.

Both classical methods for hand detection such as color based segmentation and more advanced ML methods as MediaPipe's method turned out to each have their own drawbacks and advantages. Therefore, the optimal method would most likely depend on the application.

Lastly, future studies could be carried out with a few different focuses, such as the usage of end-to-end neural networks without any image pre-processing, averaging methods over multiple frames during classification, and a good noise reduction of background subtracted image.

[1] Human–computer interaction, `https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction#Factors_of_change`, "(Online; accessed 2022-05-23)".
[2] S. Singh, Is gesture recognition hand tracking the future of touchless travel?, (2020).
[3] S. Airlines, 'contactless travel' in the new normal, .
[4] P. Viola and M. Jones, Rapid object detection using a boosted cascade of simple features, in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, Vol. 1 (Ieee, 2001) pp. I–I.
[5] D. Dahmani, M. Cheref, and S. Larabi, Zero-sum game theory model for segmenting skin regions, Image and Vision Computing **99**, 103925 (2020).
[6] e. a. Zhang, Fan, Mediapipe hands: On-device real-time hand tracking, arXiv preprint arXiv:2006.10214 (2020).
[7] R. Arya, Hand gesture recognition dataset, `https://www.kaggle.com/datasets/aryarishabh/hand-gesture-recognition-dataset`, "(Online; accessed 2022-05-10)".
[8] Z.-h. Chen, J.-T. Kim, J. Liang, J. Zhang, and Y.-B. Yuan, Real-time hand gesture recognition using finger segmentation, The Scientific World Journal **2014** (2014).
[9] C. von Hardenberg and F. Bérard, Bare-hand human-computer interaction, in *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, PUI '01 (Association for Computing Machinery, New York, NY, USA, 2001) p. 1–8.
[10] H.-S. Yeo, B.-G. Lee, and H. Lim, Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware, Multimedia Tools and Applications **74**, 2687 (2015).
[11] P. Gavrikov, visualkeras, `https://github.com/paulgavrikov/visualkeras` (2020).