Virtual Mouse Using Deep Learning

Lebanese University

Hadath Campus

Faculty of Engineering-III

2020-2021

Presented to: Dr. Mohammad Aoude

Subject: Mini-Project

Done By: Mohammad Sharara

ID: 5319

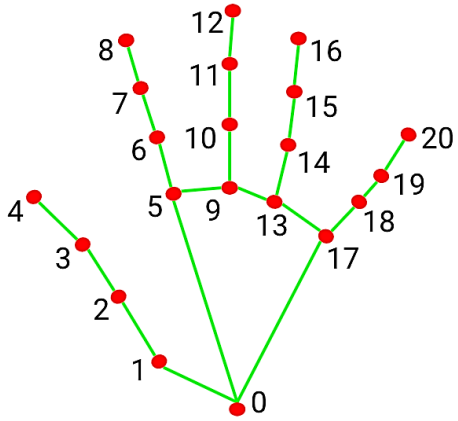
Introduction

The ability to perceive the shape and movement of the hand is an important part of enhancing the user experience of various technical fields and platforms. For example, it can form the basis for understanding sign language and controlling gestures, and it can realize the superposition of digital content and information about the physical world in augmented reality. Although it is natural for people, powerful real-time hand perception is a very challenging task for computer vision because the hands often occlude each other (e.g. finger/palm occlusion and hand shaking) and lack of high contrast pattern.

MediaPipe, a Python library developed by Google, is a high-fidelity hand and finger tracking solution. It uses machine learning (ML) to infer 21 one-handed 3D landmarks from a single frame. Although the current state-of-the-art method is mainly based on a powerful desktop environment for inference, this method achieves real-time performance on mobile phones and can even be extended to multiple hands.

Hand Landmark Model

Following palm detection throughout the whole picture, the next hand landmark model uses regression to achieve accurate key point localization of 21 3D hand-knuckle coordinates inside the identified hand areas, i.e. direct coordinate prediction. Even with partially visible hands and self-occlusions, the model develops a consistent internal hand position representation.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Fig1: Hand landmark model consisting of 21 coordinates

Training Data

To get a high accuracy, 30000 real images of hands were annotated with 21 points in 3D in order to better all the possible hand poses and provide additional supervision of the nature of the geometry of the hand. Moreover, a high quality synthetics hand model was rendered in front of various backgrounds and mapped to the corresponding 3D coordinates to further increase the accuracy of the detection of the hand against different backgrounds.

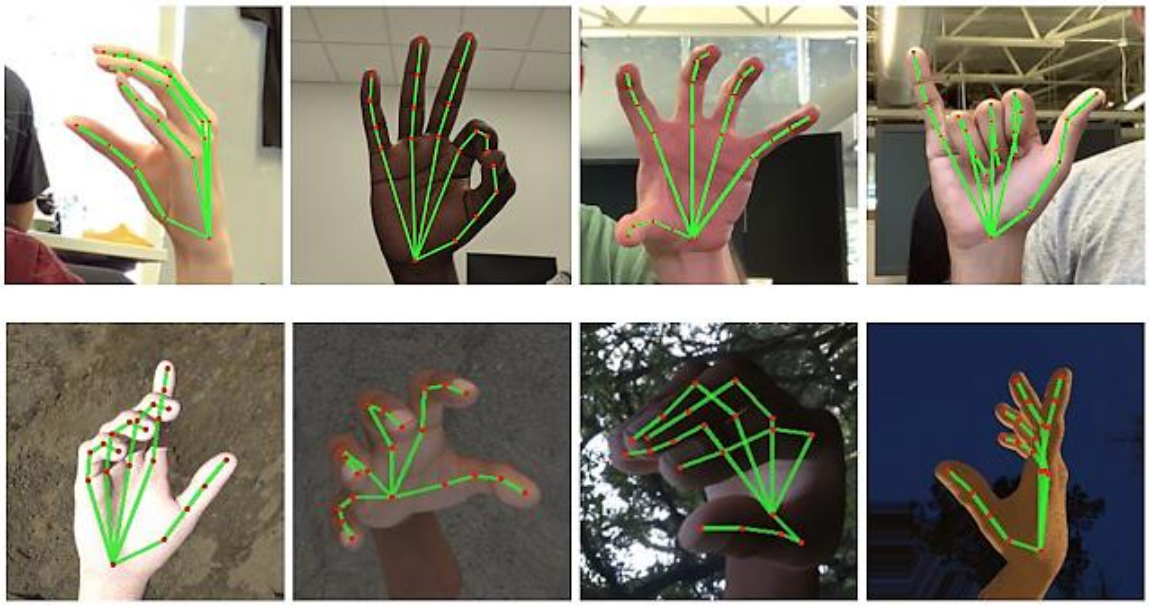


Fig.2: Several images of annotated 21 coordinates

Deep Learning Model

Using the Mediapipe library allows for abstracting the implementation details and the construction of any DL model. The neural network that was used by this library is the Convolutional Neural Network (CNN). It is composed of standard feed forward neural networks that are fully connected, i.e. each node in each layer is connected to all the nodes in the following layer. The uniqueness is that before feeding data into this network there exists two layers that filters the image and gets only the important data to be fed into the feed forward network resulting in high computational and time efficiency of processing. Thus, they are suitable for processing images since they consist

of huge quantity of data. These layers consists of filters that are small 2D matrices consisting of certain numbers and are adjusted according to the type of data you want to extract. The layers, also called kernels, will convolute over the whole image, meaning that the product will be calculated between the kernel and every 'window' or portion of the image. In this manner, only high-level data will be kept from the images and other data will be discarded. This reduces the size of the image and allow for better processing. In the below demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix. The Kernel shifts 9 times because of Stride Length = 1 (*a parameter of the neural network's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time*), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering. The filter moves to the right with a certain Stride Value until it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

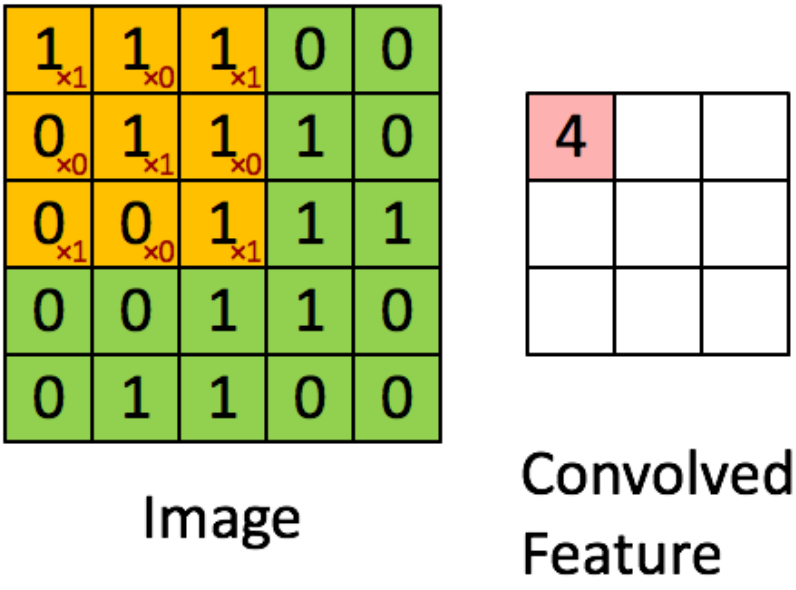


Fig.3: Kernel that is applied to the image array.

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features, which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

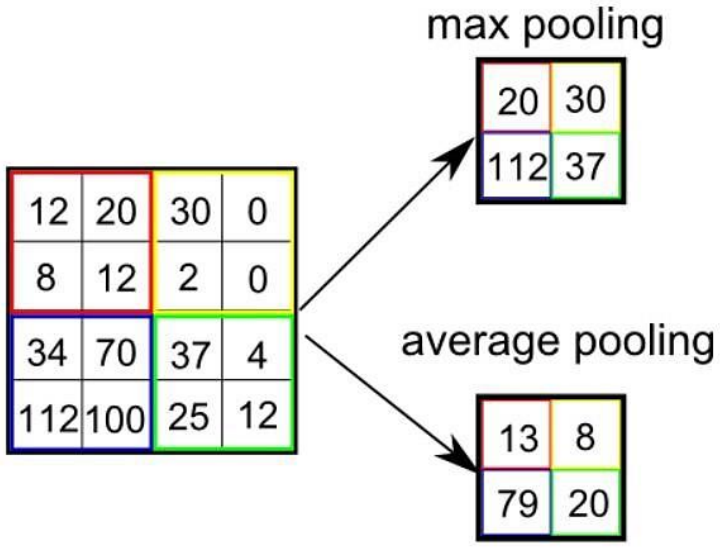


Fig. 4: Pooling types

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling. After filtering and convoluting the image array with the appropriate filter, the image array is reshaped, meaning that the dimensions of the array are adjusted to be suitable to the input layer of the network. The final stage is to feed the processed array to the network and to output the results.

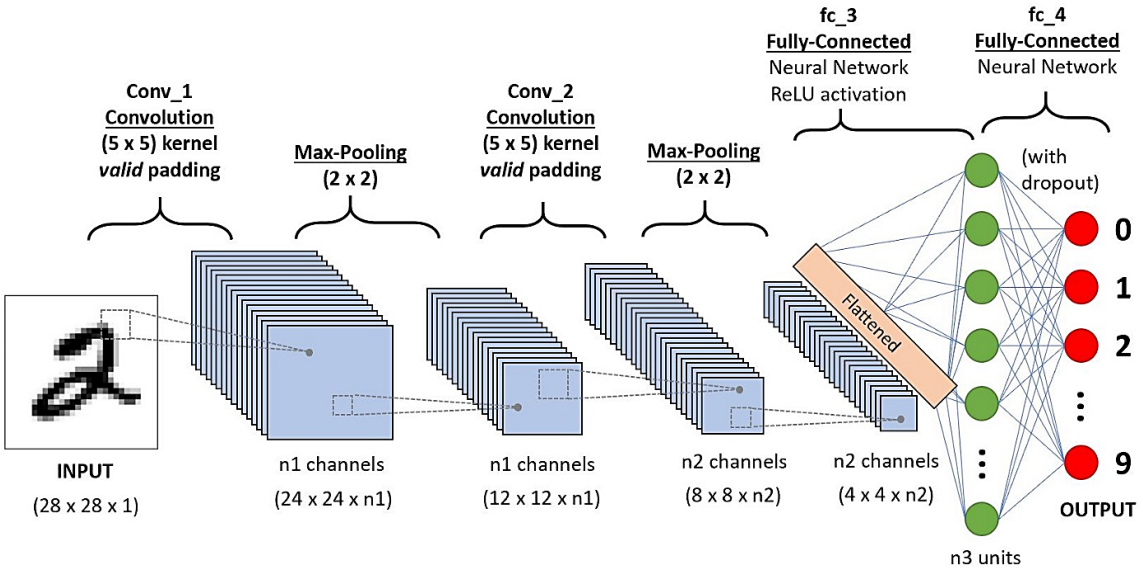


Fig.5: Diagram of CNN

Implementation

The code was implemented on Pycharm IDE and tested on the web camera of the laptop of screen resolution of 1920 by 1080 (full HD resolution). The model worked perfectly and it was possible to control the mouse pointer with ease. Of course, certain adjustments could be made concerning the smoothness of the movement of the pointer.

Conclusion

Deep learning algorithms has a lot of potential in automating and simulating human cognition. There is a high potential in this promising field to evolve and to arrive at a certain level where general artificial intelligence is achieved. Up till then, a lot of improvements should be made in this field and new technologies and frameworks will emerge to further facilitate and abstract any task. For instance, such model as Mediapipe could be improved in order to increase accessibility for those who cannot use their hands to type. Some improvements could include a modification so that one could control the mouse or even the keyboard with some gestures with ease. The overall benefit is improving user experience.