**Hand Gesture Detection for Sign Language**

Vo Manh Hieu1, Ngo Cong Minh1, Dang Thanh Tin1

1Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam  
dttin@hcmut.edu.vn

**Abstract.** For People with hearing impairment, communicate with others is such a huge problem. Therefore, Sign Language was invited to help them communicate without any sound. However, learn sign language is also a barrier due to its diversity and complexity. Nowadays, with the technology development especially in computer science, we can create a program which can read the sign language and translate it to words. It can help to reduce the time of learning a new sign language and we don’t need a translator. We applied CNN (Convolutional Neural Network) using Keras - Deep Learning API built on top of TensorFlow through python programming language and OpenCV library that can detect the sharp and movement of human's hand. Our project works by tracking hand gesture then translates to readable languages.

**Keywords:** hand tracking, sign language, Convolutional Neural Network, image classification.

# **Introduction**

Detecing the shape and movement of the human hand is crucial in improving the user experience across various technology areas. Supporting communication with people with disabilities by recognizing their sign language through hand gesture recognition is the problem that needs to be solved. The communication is continuous so that tracking hand movement in real time is required.

Before running the project, we must install input gestures via webcam and then using deep learning (CNN, Keras, tensorflow and openCV) to train completed data. After that we excute the project, with the input gesture real time, we use comparative measure to compare both real time data and trained data. Result is displayed as text on screen.

We set out to track hand gestures without having to use any more wearable devices. Running without the use of a wearable tracking device, just using the camera helps us reduce overall costs as much as possible. This is also more convenient and easier for sign language speakers and technicians, but the accuracy is somewhat dependent on the light and context around the speaker.

This paper is organized into 6 section. Section 2 is to talk about our main resources which we take as inspiration. In Section 3, we list out our training system. Specific algorithms as well as block diagrams of implementation steps are shown in Section 4, which we deepen and analyse our application. Section 5 demonstrates experimental results and comparisons while Section 6 concludes for the whole paper.

1. **Related work**

The method for the highest accuracy when solving this problem is to use sensor gloves. In 2014, Gowri. D, Vidhubala. D built a “Sign language recognition for deaf and dumb people” system and designed its own sensor gloves to recognize 26 English letters in American Sign Language with 99% recognition accuracy[1].

However, this type of device is quite expensive whether it is purchased directly or made by yourself to study the topic of gesture recognition.

In 2017, the paper Sign Language Recognition Using Kinect Sensor by Vo Hong Khanh and Pham Nguyen Khang at Fundamental and Applied Information Technology FAIR’2017 gave 99.46% accuracy[2].

This solution is generally more accessible and affordable. Even so, we still have to install special equipment called Microsoft's Kinect camera to collect information and process images.

In general, a number of studies on sign language and sign language identification have been carried out using devices with modern technologies, whether in Vietnam or in the world, identification. gestures via conventional cameras (computer webcam) have not been given much attention.

Our application is built up from existed source “Sign Language Recognition Using Hand Gestures” by Shadab Shaikh[8]. Building this app is based on the principles of gesture identification as letters, but in extension we take a sequence of frames of a complete action to form words.

# **Training system**

*Operating System:* Windows 10

*Framework:* In this topic, we use Keras - a deep learning framework based on Python and Tensorflow library[3] through Convolutional Neural Network (CNN) to conduct training. In network training CNN automatically learns values through filter classes based on how you do it. In the image layering task, CNN will try to find the optimal parameters for the corresponding filters. The last layer is used to layer the image[4]. We also use openCV as a library for computer vision, image processing and machine learning[6]. We chose it because this is an open source library with a BSD license that allows us to use for free for academic purposes.

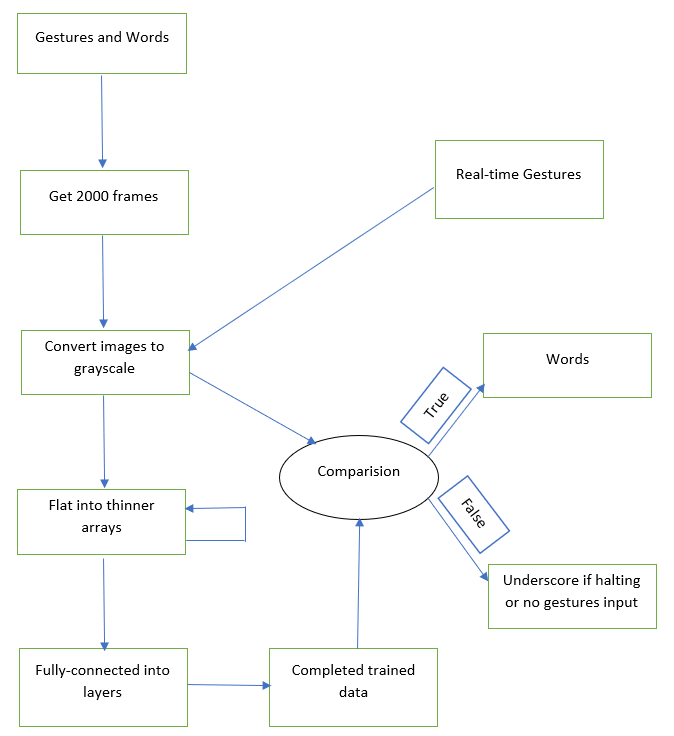
*Graphic User Interface:* Pyqt5

*Deployment method:* Pyinstaller

*Hardware:* CPU core i5-7200U 2.5GHz ; RAM 8GB ; HD Webcam

1. **Algorithm**
   1. **Flowchart**

The main idea here is to build the data that the machine has learned and then compare the real-time data with the learned data.



**Fig. 1.** Flowchart of main program.

* 1. **Collecting date and image processing**

To build the data set as a model and training we choose Vietnamese sign language. We opted out of the Ho Chi Minh Sign Language Dictionary for 12 words of sign language[9]. Each word in the archive is recorded with a total of about 2200 samples. These gestures are captured using the camera.

After that, the best 2000 samples were selected. All of these images are extracted from the full animation of gestures, from the first frame to the last frame. The recorded sample will be processed to remove the background and keep the hand grayscale image.



**Fig. 2.** Processed samples.

For more detail, we restricted an area of the image captured by the camera, cut out that part and processed the color through the OpenVC library with “cvtColor” and “bitwise\_and”.

* 1. **Training**

To start, we learn the training principle and we choose to convert a gesture sequence into multiple split frames and learn them in an orderly manner. Learning the frames broken down from an action will help maximize the machine's processing / learning capabilities and to learn these frames we choose the CNN training method through Keras.

To reduce the number of parameters we need to learn. We only need to learn the weights of the filter (which usually is a lot smaller than the input image) instead of learning the weights connecting each input pixel.

CNN method is one of advanced deep learning paradigms. Especially, CNN is widely used for solving object detection of images. The CNN is divided into 3 dimensions: wide, high and deep. In the network, the neurons are not fully connected to the entire next neuron but they still have a relationshop. Thus, an output layer is minimized to the vectors in those 3 dimensions.

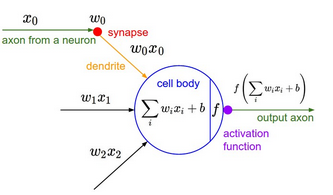
A picture containing map, text

Description automatically generated

**Fig. 3.** Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer[7].

**Cnn\_model.py will train our data. Here are 4 steps pre-train (create a completed CNN).**

*Step 1.* Initialing sequentially the images, it’s encoded by classifier.add and Convolution2D with 2 values 0 (black) and 1 (white) into the size 64×64×3, convolution mean every element is multiplied with the matrix 3×3 and sum up into 1 cell in new matrix that we call convolved feature. The convolved feature has the size 64×3×32. The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially. Finally, we have 3 convitional layers with the recognition feature.



**Fig. 4.** Convolution in Network but the neurons chapter remain unchanged[10].

*Step 2.* Here we use Maxpooling method with Maxpolling2D. The pooling layer is used right after the convulational layer to simplify output information to reduce the number of neurons. The pooling procedure is applied as max-pooling, which selects the maximum value in the 2×2 input area. Thus, through the Max Pooling layer, the number of neurons is halved. So every time we add a convolution layer, we apply maxpooling. In this model, we use a total of 3 convitional layers.

*Step 3.* Before fully connect among layers, we flat them with classfier.add(flatten())

*Step 4.* Finally, we put all the layers together into a completely CNN. 2 final layer is a fully connected layer. This layer connect every neuron from max-pooled layer to every output neuron. At this point, we have a CNN used for training.

**After creating a CNN, we jump to training part with compile() method.**

*Step 1.* We specify the training configuration (optimizer, loss, metrics) by classifier.compile

*Step 2.* We call fit(), which will train the model by slicing the data into "batches" of size "batch\_size", and repeatedly iterating over the entire dataset for a given number of "epochs". The Keras library will help retraining our CNN. At this point, we have a completely trained model[5].

All the models we trained are converted by classifier.save into h5py file (VSLModel.h5) which is a lot lighter than the number of frames input that we put in, just encapsulated in one model file.

* 1. **Recognition of Real-time Data**

The problem when we compare real time is that we need to compare data sources, so we use deep learning to do that. As we mentioned the above image processing method, we also use it for handling the input gestures. We use OpenCV for the webcam recognition. The input gestures are converted into grayscale mode.

A screenshot of a cell phone

Description automatically generated

**Fig. 5.** Single word detection in real-time

At this point, we exploit the comparison method of Keras (Tensorflow framework) which can connect with trained data. Our real-time input has size of 64×64, we used img\_to\_array to encode it into matrix of 0 and 1 and into vectors then we start to compare with our trained model (VSLModel.h5) by classifier.predict. With the accuracy accepted, the result text is appeared in the screen. However, we set up the gap among words based on the delay of our input gestures. For example, if we don’t make any operation or we halt our action for a moment, the program understand it as a delay and the output will be an underscore. The resting frame we set here is 20. We also make a create gestures option which can allow to create your own words but only with low accuracy (1 image to compare) in 1 session.

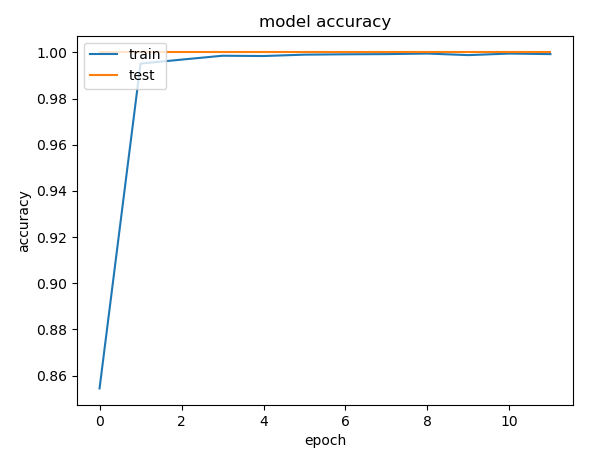
Our last part is to make an application that everyone can use easily so we use PyQt5 to design a Graphic User Interface.

Finally, we execute our project with Pyinstaller. Furthermore, we pack it to 1 file installation which people can use it easily even without installing Python.

1. **Experimental result**

For single gesture tested, we received high accuracy result. Therefore, it is supposed to say that the trained data has worked properly. This is real-time testing, with the latancy less than 0.5s. However, the brightless afftected a lot in the results since we must convert real-time gestures into grayscale. Furthermore, our project has a limitation which can allow the gestures with the clean background only. If we have a complicated background, for example, our body, the real-time input is disturbed.

For the training accuracy test, we captured 200 frames of a real-time gesture and start to test the accuracy compare with data trained, the accuracy oscillate between 99% and 100%, which can show that our training method has a very high accuracy. Our model is focusing a lot of attention on training data and does not bring generality on never-before-seen data. This leads to the model achieving extremely good results on the training data set.



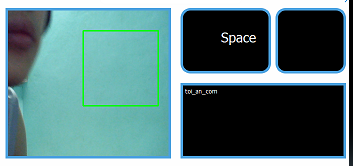
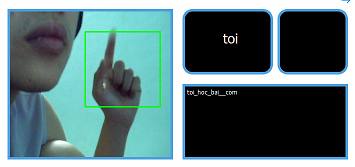
**Fig. 6.** Training accuracy plot using matplotlib library.

In manual testing, we perform 10 gestures in 200 times and record the obtained results and calculate to percentage accuracy, the accuracy oscillate between 97% and 99.5%. This test is performed on only one camera, the actual accuracy may vary with poor camera quality.

**Table 1.** Result in manual testing

|  |  |  |
| --- | --- | --- |
| Word | Correct | Accuracy |
| Cơm | 198 | 99.0% |
| Đi | 198 | 99.0% |
| Không | 197 | 98.5% |
| Học | 198 | 99.0% |
| Tôi | 197 | 98.5% |
| Có | 194 | 97.0% |
| Bạn | 199 | 99.5% |
| Bác sĩ | 195 | 97.5% |
| Ăn | 198 | 99.0% |
| Ngủ | 198 | 99.0% |

For the sentence mode, the project has worked with the accuracy around 95% (statistics 200 tests). We face the problem moving from one word to another, which is the set up of delay recognition. If we do too fast, there is no underscore among words, otherwise if we do too slow, there are a lot of underscores among words. The resting frames are 20 so its quite sensitive.



**Fig. 7.** “toi an com” and “toi hoc bai” recognition.

Overall, the application works in real-time smoothly and acceptable. However, we are doing with the low trained data. With the increase of trained data, the accuracy is changed. We will work more to handle all the existed problems. Thus, we will try to go a step further with the addition of background and background filtering.

1. **Conclusion**

After doing research in deep learning and computer vision, we have created a simple, inexpensive application to track hand gestures for translating sign language into plain writing. The program uses conventional cameras to recognize sign language, therefore optimizing costs and simplifying installation and application process. In this project we only sample a certain number of words due to the limitations on our computer hardware used for deep learning. This also affects usability as the trained data is not yet extensive. Of course the most obvious application of this program is that it helps a lot for sign language translation. This program can also be developed to be applied to pre-recorded videos to make content from disabled people more accessible. The program can be applied in communicating with users of sign language in a more natural way if integrated into portable devices such as smart phones, smart glasses. If used in conjunction with video communication programs, it will be more convenient. The project can expand the sign language of countries around the world, thereby helping the mute and deaf have more opportunities to interact with foreigners. Without limiting sign language recognition, programs can be developed to recognize other hand gestures, thus being applied in many other areas such as communicating with a computer through gestures, Artificial Intelligence… The app works effectively only under certain environmental lighting conditions and this looks rude and primitive at first. However, we strongly believe that in terms of future prospects.

**References**

1. Gowri.D, Vidhubal.D (2014). “Sign language recognition for deaf and dumb people”. Int J Res Eng Technol, 03(19), pp. 797–799.
2. V. H. Khanh, P. N. Khang, "Sign language recognition using Kinect sensor", FAIR’2017 Fundamental and Applied Information Technology.
3. “Keras”, https://keras.io/about/, last accessed 2020/07/30.
4. “Convolutional Neural Network - MATLAB & Simulink”, https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html, last accessed 2020/07/30.
5. Andrew Gibiansky, “Convolutional Neural Networks” [Online]. Available: https://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/, last accessed 2020/07/30.
6. “OpenCV”, https://opencv.org/about/, last accessed 2020/07/30.
7. “Deep Learning - MATLAB & Simulink”, https://www.mathworks.com/solutions/deep-learning/, last accessed 2020/07/30.
8. Shadab Shaikh, “Sign Language Recognition Using Hand Gestures”, https://github.com/shadabsk/Sign-Language-Recognition-Using-Hand-Gestures-Keras-PyQT5-OpenCV, last accessed 2020/08/05.
9. “Vietnamese Sigh Language Dictionary”, https://tudienngonngukyhieu.com/, last accessed 2020/08/08.
10. CS231N Staff, Stanford University, “Convolutional Neural Networks for Visual Recognition”, https://cs231n.github.io/convolutional-networks/, last accessed 2020/08/08.