VNU - HCM - University of Science

FINAL REPORT

Python for Data Science

Lecturer: Ha Van Thao

Members:

20280066 - Tran Le Minh

20280028 - Le Thi My Hang

20280064 - Mai Thi Thao Ly

20280088 - Nguyen Thi Hong Thi

December 22, 2022

SIGN LANGUAGE DETECTOR

GROUP 8

Work Assignment

No.	Name	Student ID	Work Assignment
1	Tran Le Minh (leader)	20280066	• Work progress management • Making model (processing data, ...) • Main ideas for .ipynb webcam file • Design GUI app python • Writing report
2	Le Thi My Hang	20280028	• Writing report • Writing .ipynb file report with markdown • Find bug and optimize the code • Contribute ideas
3	Mai Thi Thao Ly	20280064	• Plan an idea • Find data set • Design GUI app python • Working on design web app • Contribute ideas
4	Nguyen Thi Hong Thi	20280088	• Design GUI app python • Clean data • Working on design web app • Contribute ideas

Note: All of the members have equal contributions, some of the parts of the contributions was failed or not used in the final project.

Contents

I Introduction **3**

II Description of Overall Software Structure **3**

III Data **4**

1 Dataset 4

2 Data processing and cleaning 4

2.1 Contour Edge and Canny Edge detection 4

2.2 Landmarks 5

IV Model **6**

1 Convolution Neural Network 6

2 Dense Neural Network 6

3 Raw Data and Edge detection Data 7

4 Landmark Data 8

V GUI Webcam **9**

VI Conclusion **10**

I Introduction

Deaf or dumb people use Sign Language as their main communication tools, so we - normal people - can not understand what they say if we don't study Sign Language. So one of my friends comes up with an idea that we create a neural network that can detect Sign Language. Sign Language use hand and body movement to express what they want to say. However, there are a lot of different types of Sign Language and body gestures so it's hard for a simple neural network to learn and process them all. By using Deep Learning Algorithms and Image Processing we can classify hand gestures into text. So we select American Sign Language as our main Sign Language for this project.
Our selected dataset has over 80000 images from 29 classes. With 26 classes belonging to the alphabet, one is Delete and the other is Space and Nothing.
Although Convolution Neural Network (CNN) is widely used for Image/Video processing, we only use Dense Neural Network (DNN) with the help of mediapipe hand processing.

II Description of Overall Software Structure

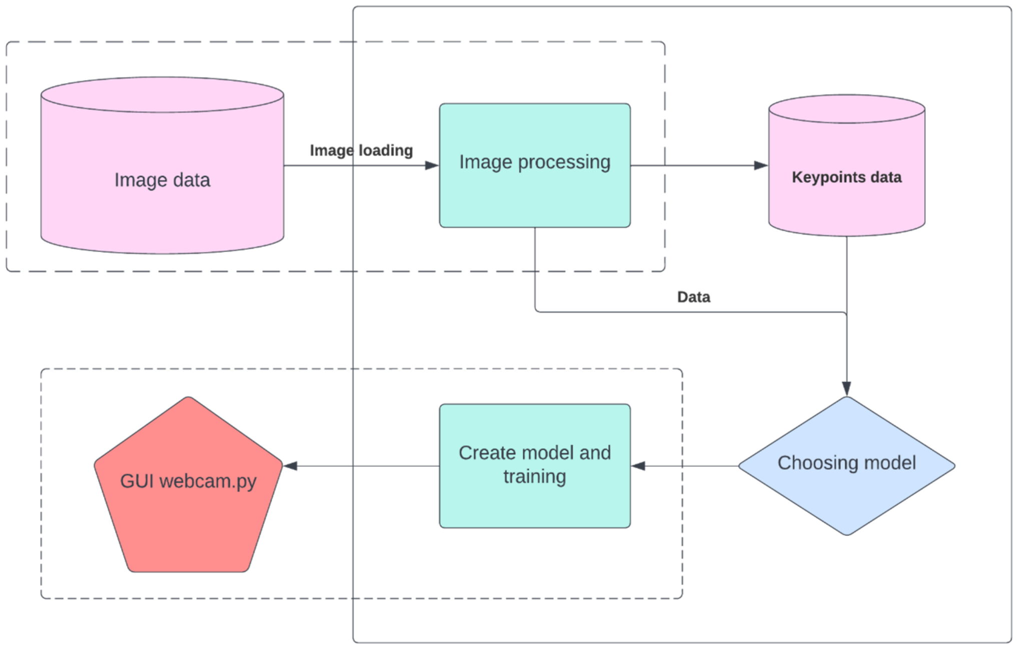


Figure 1: Diagram of project structure

As shown in Figure 1, the project was divided into 3 parts:

- Data processing: Data was loaded in and processed. We tried a lot of types of model so there were lots of types of image data after being processed.
- Training: Different types of model was trained in order to archived the best model.

- Create GUI webcam app.

III Data

1 Dataset

We only use the American Sign Language dataset. Because the American alphabet and English was widely used around the world.
The dataset is a collection of 87000 200x200 pixels images of the alphabet from American Sign Language, separated into 29 folders that represent the various classes (previously describe). Then we flip the image to get the left hand data.

2 Data processing and cleaning

We consider the Nothing in dataset was redundant so we remove it. At this time we have 2 option for data processing:

- Use Contour Edge and Canny Edge detection methods to processing image.
- Use Mediapipe Hands Model to get landmarks dataset.

2.1 Contour Edge and Canny Edge detection

An image contains 3 channels (RGB) with each channel having 200x200 pixels which are ranging from 0 to 255 in integer or 0 to 1 in float. 0 means black and 255 means white and so on with float-type data.
In Contour Edge and Canny Edge detection will reduce the channels down to 1, and then detect a wide range of edges in the images. With Canny Edge detection, the edge of an object is clearer than with the Contour Edge detection method. After we run Canny Edge detection on the image, we then combine it back to 3 channels by duplicating 1 channel to 3 and we have the results.

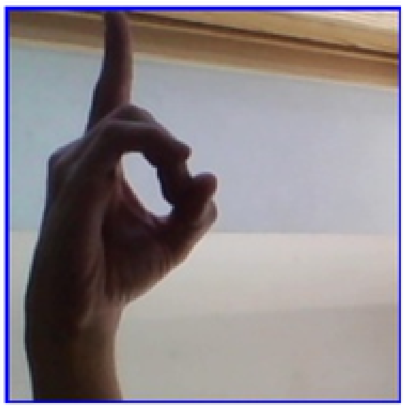


Figure 2: Before Edge detection

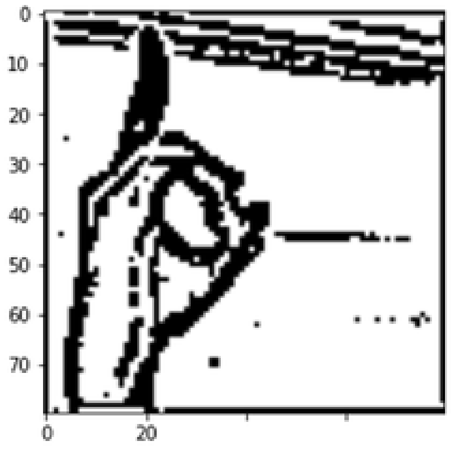


Figure 3: After Edge detection

The steps we have taken on this way of processing image:

1. Read Images.
2. Convert Images to Gray scale.
3. Run throw GaussianBlur with kernel size (5,5) to reduce noise on the image. The equation for a Gaussian filter kernel of size $(2k+1)(2k+1)$ is given by:

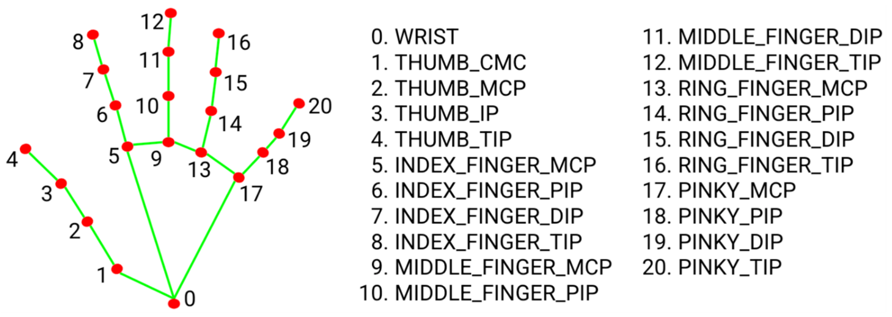
$$H_{ij} = \frac{1}{2\pi\sigma^2} exp - \frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}; 1 \leq i, j \leq (2k+1)$$

4. Run image throw adaptiveThreshold and Threshold (double Threshold) to increase strong pixel gradiant value.
5. Resize image to 80x80 pixels and return 1 channel image .
6. Stack 3 channel and the image is now 80x80x3.

In conclusion, Edge detection works really well in order to get the edge of the object if the background of the image was smooth. If the image have too much noise then Edge detection can not get the edge very accuracy.

2.2 Landmarks

After some fail with Edge detection we come up with the idea of using hands landmarks. In landmarks detection we have several choices such as YoloV7, OpenPose,... but we choose Mediapipe. Mediapipe was developed by Google so it's performance way better than other.
We use Mediapipe Hands model to process image. Mediapipe Hands model return a results variable that contain 21 points on your hand, with each point have 3 values (x, y, z), and different types of information about that hand.



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

Figure 4: 21 hand landmarks

Collection of detected/tracked hands, where each hand is represented as a list of 21 hand landmarks and each landmark is composed of x, y and z. x and y are normalized to [0.0, 1.0] by the image width and height respectively. z represents the landmark depth with the depth at the

wrist being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x.
But the Mediapipe Hands detection model only works well when it sees hand clearly, which means if the image is too dark or too small, or too blur it can not detect your hand very accurately or do not detect it at all.
The steps we have taken on this way of processing image:

- Read Images.
- Increase images brightness.
- Let it run throw Mediapipe Hands model.
- Extract keypoints.
- Remove none detectable landmarks by put it in Pandas dataframe and remove full zeros row.

IV Model

1 Convolution Neural Network

Computer Vision is a field of Artificial Intelligence that focuses on problems related to images and videos. CNN combined with Computer Vision is capable of performing complex problems.

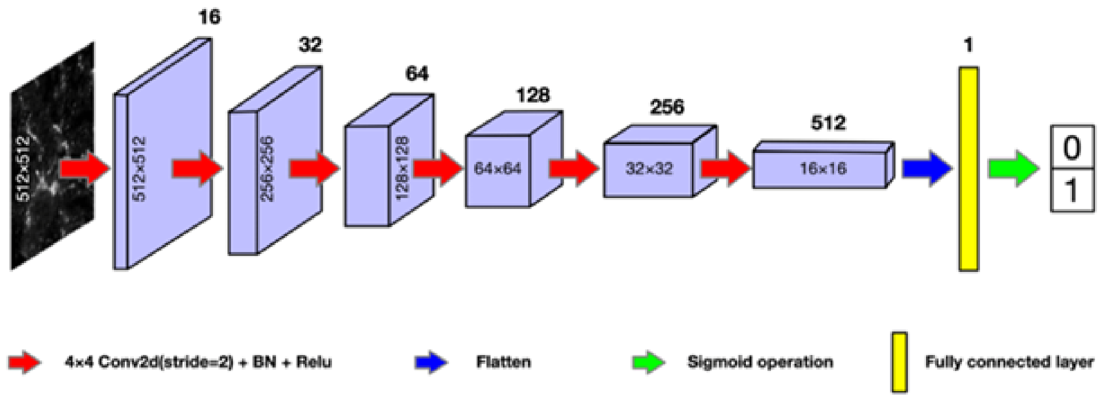


Figure 5: Example of a simple CNN

When we approach CNN model we have tried different types of data processing, from raw data to Edge detection.

2 Dense Neural Network

Dense Neural Network is one of the most straightforward configurations for a deep learning model, a densely connected neural network. This is typically a model that will achieve the highest performance on text data. In our project was the same of reading text data by reading letters.

In order to build a good working dense model we need to understand the rules of building dense layer. Because the size of the next dense layer had to be smaller than the previous one in a densely connected neural network.

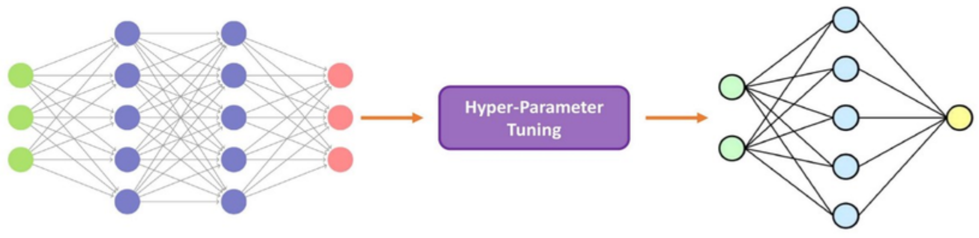


Figure 6: A simple DNN structure

3 Raw Data and Edge detection Data

We build a simple conv2d CNN with input layer as 80x80x3. (figure 6 below)
Model accuracy was 97% but when we test with the test data from Dataset, some of the images it detect were wrong. Such as the letters A, E, and S and the letters M, N, and some other letters. Also when we test with our dataset it's always wrong.
So we decide to use prebuild model on tensorflow. We test 2 model, 1 is ResNet-50 and the other is EfficentNetV2M. Both of them we add 1 Flatten Layer and 3 Layer of Dense, 1 Dropout Layer and 1 Final Dense Layer.
These 2 model accuracy was 94% and when we test with the test data from Dataset, it detect very well, everything was correct but when we test with our dataset it was always wrong. Although we process our image like what we done with the train data.

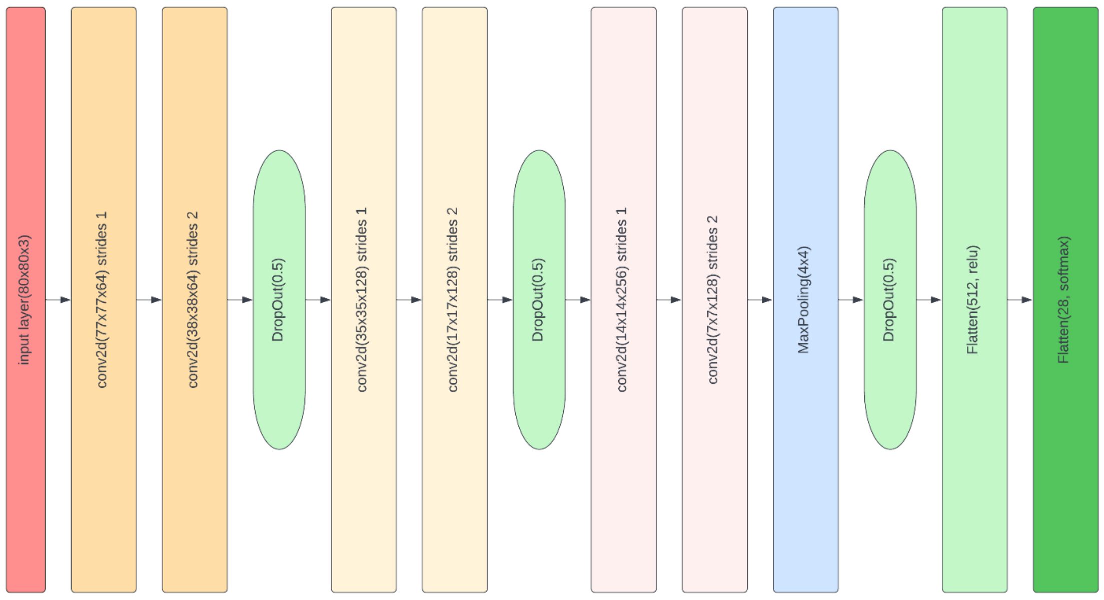

input layer(80x80x3)
conv2d(77x77x64) strides 1
conv2d(38x38x64) strides 2
DropOut(0.5)
conv2d(35x35x128) strides 1
conv2d(17x17x128) strides 2
DropOut(0.5)
conv2d(14x14x256) strides 1
conv2d(7x7x128) strides 2
MaxPooling(4x4)
DropOut(0.5)
Flatten(512, relu)
Flatten(28, softmax)

Figure 7: Our CNN structure

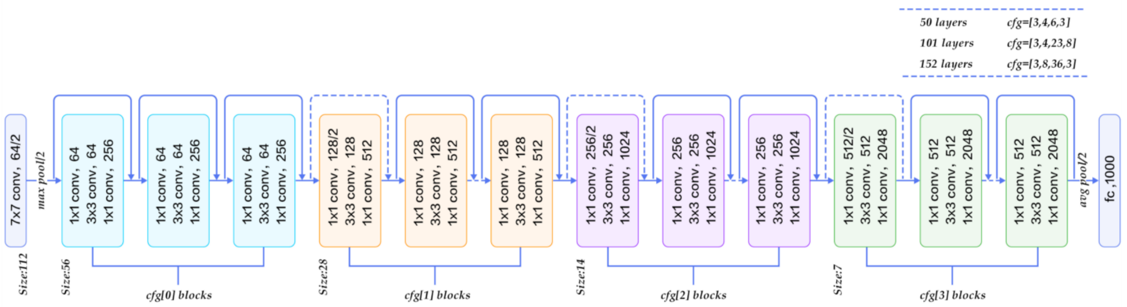

50 layers cfg=[3,4,6,3]
101 layers cfg=[3,4,23,8]
152 layers cfg=[3,8,36,3]
7x7 conv, 64/2
max pool/2
1x1 conv, 64 3x3 conv, 64 1x1 conv, 256
1x1 conv, 128/2 3x3 conv, 128 1x1 conv, 512
1x1 conv, 128 3x3 conv, 128 1x1 conv, 512
1x1 conv, 256/2 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 256 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 512/2 3x3 conv, 512 1x1 conv, 2048
1x1 conv, 512 3x3 conv, 512 1x1 conv, 2048
avg pool/2
fc,1000
Size:112
Size:56
Size:28
Size:14
Size:7
cfg[0] blocks
cfg[1] blocks
cfg[2] blocks
cfg[3] blocks

Figure 8: ResNet-50 structure

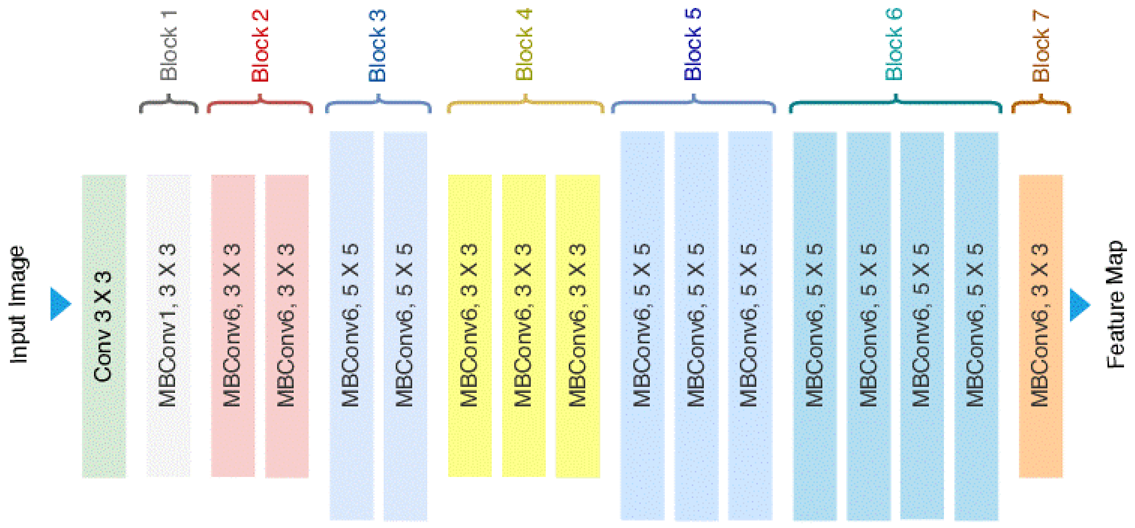
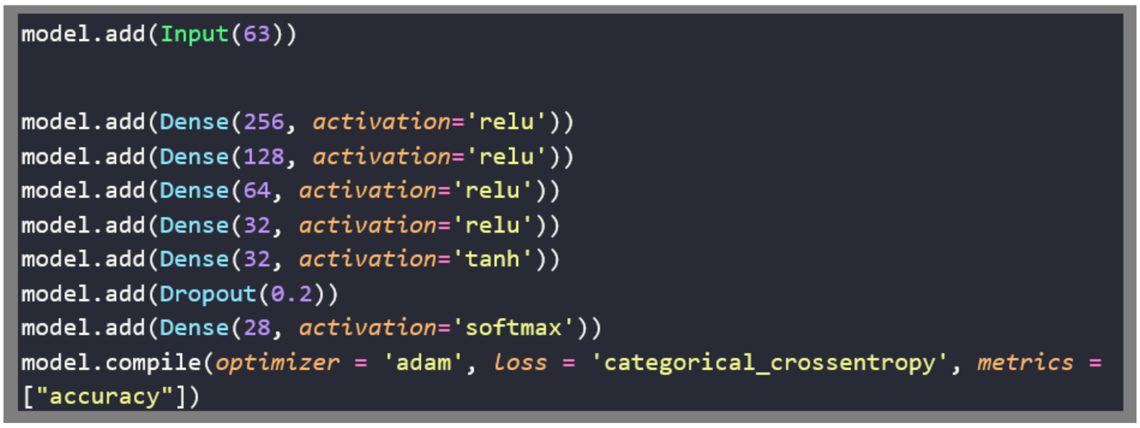


Figure 9: EfficentNet B0 structure

4 Landmark Data

With Landmarks data we first use conv2d and conv1d to build a model but the result was not very good. Again the test data from the dataset was perfect but with our data set the accuracy was about 60%. We also tried to normalize the landmarks but the outcome still the same. So we decide to build a very simple Dense Neural Network.



```
model.add(Input(63))


model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(28, activation='softmax'))
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
["accuracy"])
```

At first we do not add a Dense tanh layer and the results still good but it's confuse between letter

N, M and Delete gesture. After we add Dense tanh layer in and this time the result was fantastic. We manage to archive val_accuracy up to 99, 24%

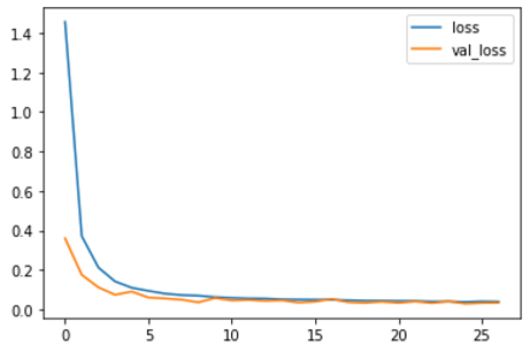


Figure 9: Traing and Validation lost

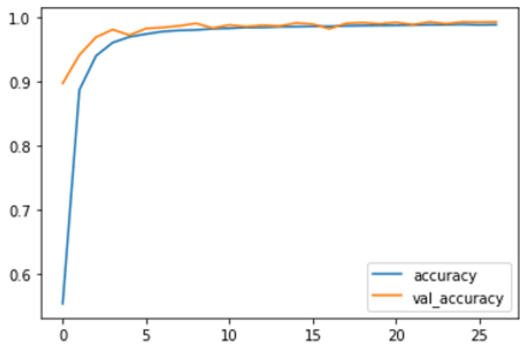
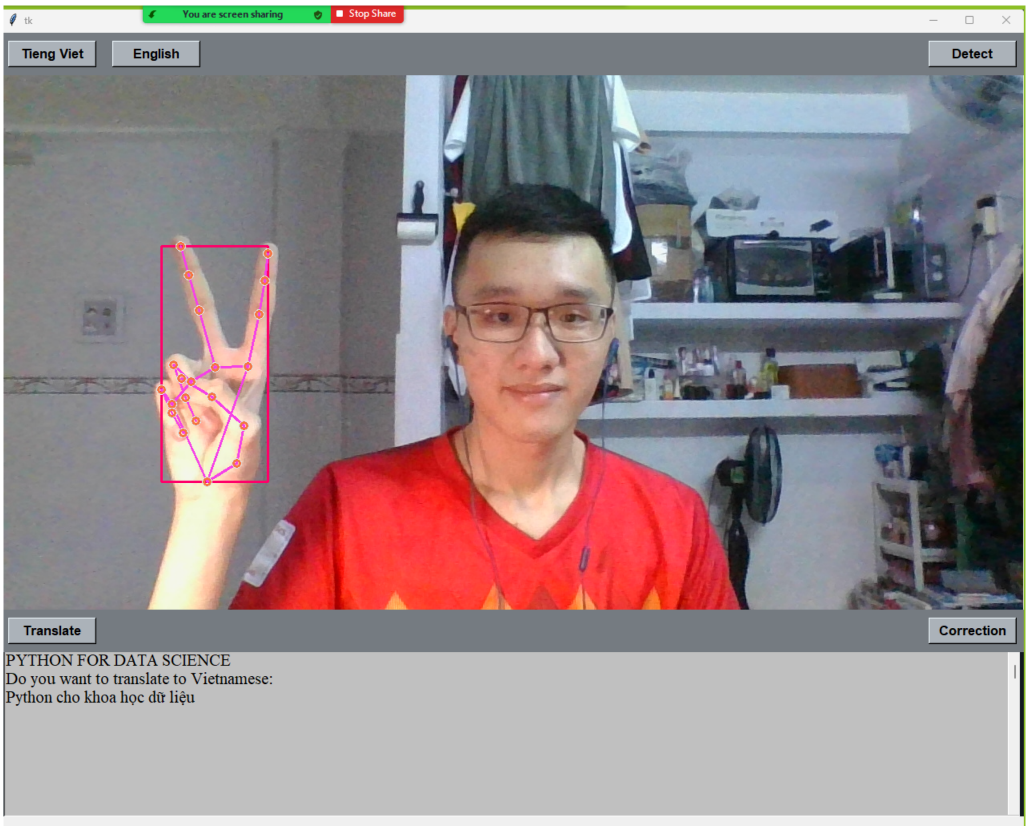


Figure 10: Training and Validation accuracy

This model is very good at predict the correct letter. With letter that have similar shape like V and W or A, E and S, or M, N or Delete gesture. it's still work very fine. Note: It's only predicts rights if you wave your hand right, if not it's may still wrong.

V GUI Webcam

After we build the model successfully, we then build a GUI Webcam app to run the model and output the text corresponding.



PYTHON FOR DATA SCIENCE
Do you want to translate to Vietnamese:
Python cho khoa học dữ liệu

Our webcam app has some other feature other than reading Sign Language:

- Selecting between Vietnamese and English.
- If selected English than you can translate it to. Vietnamese or if you type it incorrectly you can correct it if you want.
- If Vietnamese was selected than other features can't be used.

We use Tkinter to design GUI app, translator library to translate Eng to Vie, SpellChecker library to check for error.

VI Conclusion

In conclusion, we were successfully able to develop a practical and meaningful system that can able to understand sign language and translate that to the corresponding text. There are still many shortages of our system, like this system can only detect A-Z alphabets hand gestures but does not cover body gestures and other dynamic gestures. We will continue to develop this project.

References

[1] Justin Chen, 2017, Sign Language Recognition with Unsupervised Feature Learning.

[2] Anna Deza and Danial Hasan, 2018, Sign Language Recognition.

[3] D. Metaxas,June 2011, Sign language and human activity recognition, CVPR Workshop on Gesture Recognition.