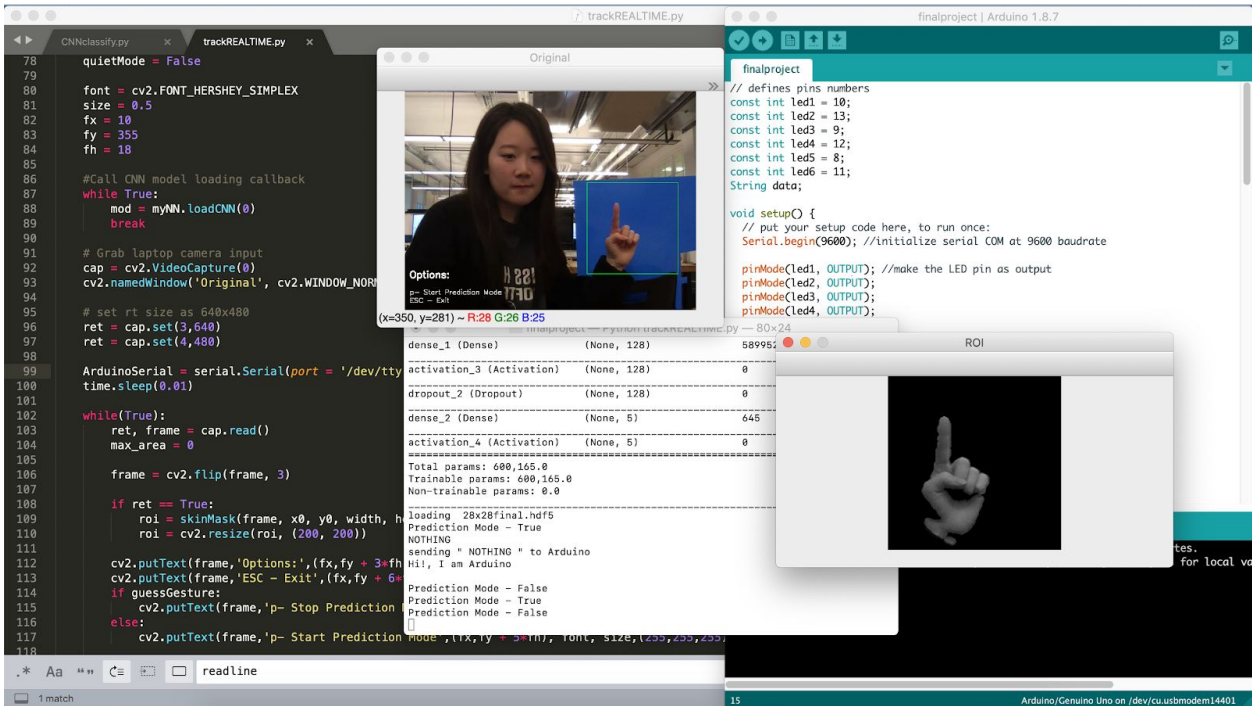
ART AND MACHINE LEARNING
CMU 2019 SPRING
FINAL PROJECT

Real-Time ASL to Braille



Stella Kim (hyunjunk)
Kevin Song (kmsong)
GitHub:
https://github.com/stellahyunjungkim/ArtML_FinalProject

Introduction

"Art is a language, an instrument of knowledge, an instrument of communication" (Jean Dubuffet). Our team also believes the converse to be true; language is a form of art. However, this luxury of communication is not always accessible to everyone.

Consequently, our team was interested in the possibility of utilizing machine learning to primarily assist in real-time American Sign Language recognition. Our secondary goal was to create a device that would allow a person who utilizes ASL to communicate with a blind person.

Fig 1. Digits in American Sign Language

→

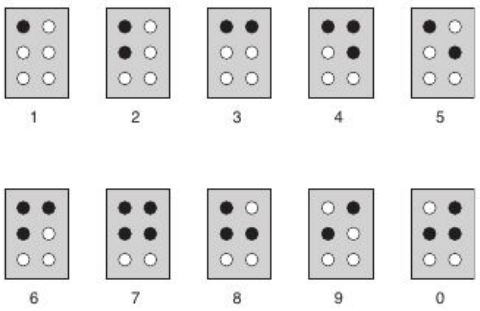


Fig 2. Digits in Braille

Approach

A. Capturing and Preprocessing Image

1) Capturing the Hand Gestures

We used OpenCV to create a simple user interface in capturing the hand gestures. Once the system is up and running, the user will see the Webcam's view of themselves on a window that pops up. We have boxed out a portion of the Webcam's view which will be captured and passed into the neural network and classified. This portion is shown by a green square on the user interface window. The image in the green square is captured once every 0.01 second and sent over to be preprocessed.

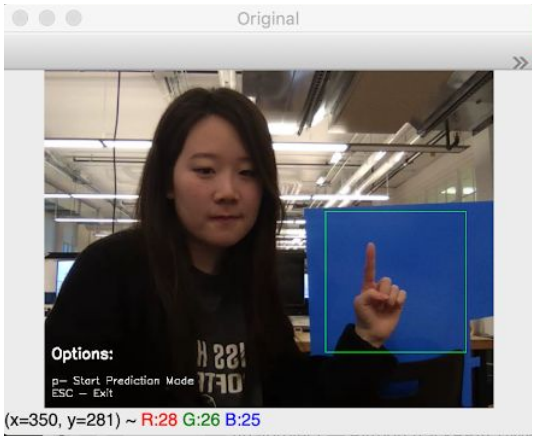


Fig 3. User Interface

2) Preprocessing the Images

The captured image is run through two preprocessing steps: skin-masking and resizing. In the skin-masking stage, the image is converted into grayscale and basically differentiates between the hand (skin) and background images. Then, it is rescaled to 28x28 pixel which is the input image size for CNN.

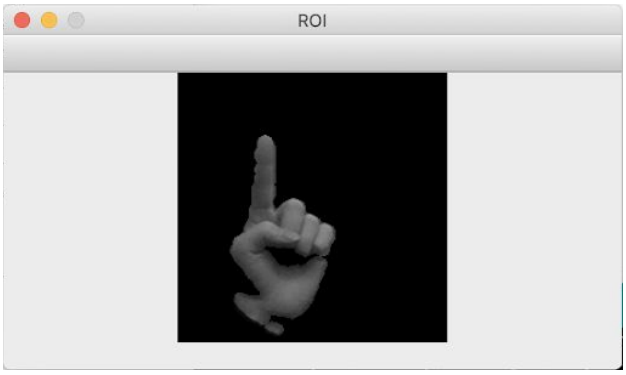


Fig 4. Preprocessed image

B. Convolutional Neural Network

1) The Basics of Neural Network

Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is a probability of a classification.

Detailed model of CNN is shown in Fig. 5.

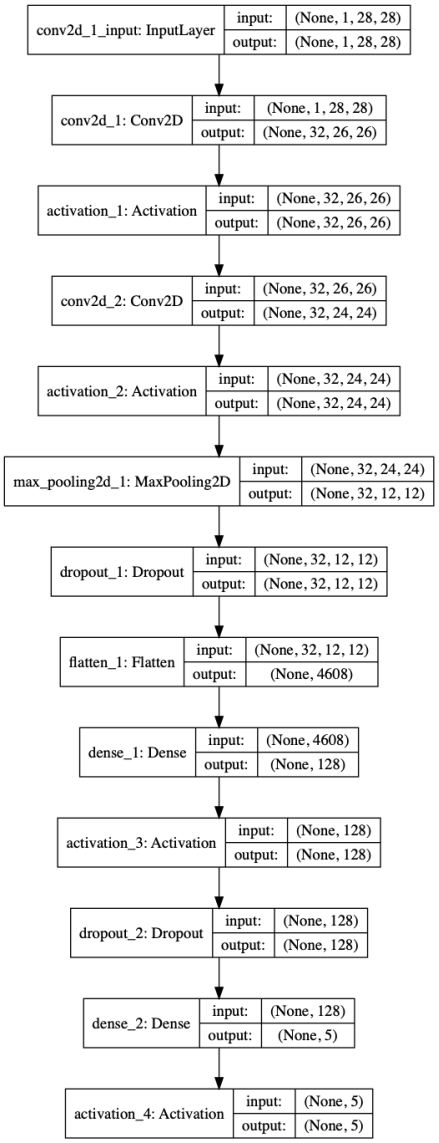


Fig 5. CNN model

2) Dataset

Since we are not provided with any of the dataset, we needed to collect enough samples and make our own dataset. We collected 1000 samples for each gesture (0-9 digits in sign language and 'nothing') of certain distance and scale. We chose to make the image size to be 28x28 pixel, since greater quality of image takes much longer time to train. We figured out the 28x28 image size is enough for CNN to discriminate each one. It took about 5 hours to train.

For measuring the accuracy of our CNN, we used K-cross validation method. From 10000 samples, 7500 are used to train and rest of 2500 samples are used to validate. We had 20 epochs in total to reach higher accuracy. Accuracy graph is shown in Fig. 7.

C. Arduino

1) Receiving data from CNN

We used pyserial module to make connection between arduino and CNN. Arduino reads the classified output generated from CNN on our laptop through the serial connection at 9600 baud rate. Based on the received data, it lights up the corresponding LEDs.

2) LED wiring

Six LEDs that represent each pin of 6-dot braille are connected to six different digital I/O pins on arduino. 220 ohm resistors are used to control the current through the LEDs.

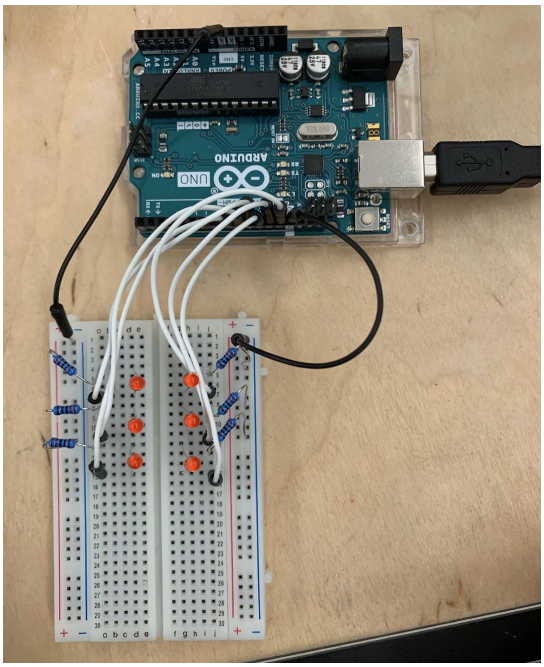


Fig 6. Arduino circuit wiring

Our Results

A. Accuracy

As shown in Fig. 7, we achieved a training accuracy of 0.9895, a training loss of 0.0361, a testing accuracy of 0.9949, and a testing loss of 0.0147.

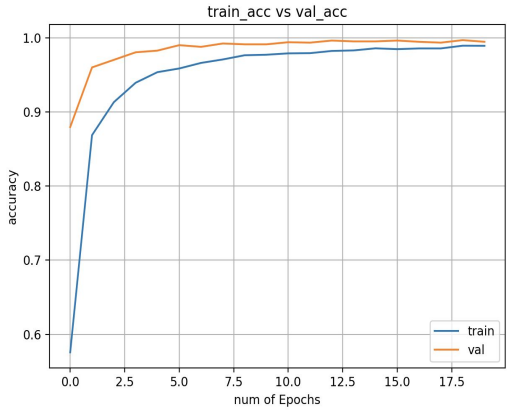


Fig 7. Accuracy of CNN classification

B. Visualization

We managed to successfully transfer the data over to arduino, and light up the corresponding LEDs. However, there is a bit of delay (~0.5 seconds) until the data is completely sent over through the serial port.

We also built a box made out of semi-transparent material to cover up the wires and see the braille representation more clearly.

Reflection

Our team successfully trained a CNN network to recognize static ASL gestures, specifically numerical digits 0-9. Furthermore, we also successfully created a conceptual prototype for ASL to braille conversion. Due to our limited resources, we were unable to create a working braille machine, and thus created a proof-of-concept model using LEDs to represent the correct corresponding braille patterns. With more resources, our team hopes to create a fully functioning real-tim ASL to braille machine that could facilitate communication to those with hearing and vision impairments.

References

1. http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html
2. https://github.com/asingh33/CNNGestureRecognizer