Capstone Project 2 – Analyze Hand Gestures
Milestone report 1

The chosen project aims to analyze and create CNN model for recognizing 6 gestures.

Since the CNN requires a high number of jpgs to train the network, I decided to get the jpg images as frames from a video of myself in front of a white backdrop. In this video, I am showing signs from 0 to 5, in different positions and different distances from the camera, for about 2 minutes each. These signs are converted into jpg images.

Data cleaning is being done by eliminating the frames that are temporally positioned between the two successive signs. Furthermore, I am deleting the images that are cut by one of the edges of the screen and the ones that are hard to understand due to various reasons.

After moving train and test data to corresponding classes, there are:

- 12460 training jpgs (and 7 directories, total = 12767)
- 5331 test jpgs (and 7 directories, total = 5338)
- 1.6 GB used for training jpgs
- 708 MB used for test jpgs

The number of files for each of the 6 classes in /train and /test is relatively equal with the exception of the classes 4 and 5 where I had to delete more jpgs due to recording issues.

These are examples for the 6 classes:

I am using tensorflow 2.0 and ImageDataGenerator returns the following strings for /train and /test folders:

/train: “Found 12460 images belonging to 6 classes.”
/test: “Found 5331 images belonging to 6 classes.”

These values are identical to the ones counted from the operating system.

The 6 classes are encoded by using one-hot encoder. Examples:

```
Image 1  one-hot encoded class: [0. 0. 0. 1. 0. 0.]
Image 2  one-hot encoded class: [0. 0. 0. 1. 0. 0.]
Image 3  one-hot encoded class: [0. 0. 0. 1. 0. 0.]
Image 4  one-hot encoded class: [0. 1. 0. 0. 0. 0.]
Image 5  one-hot encoded class: [0. 0. 0. 1. 0. 0.]
```

ImageDataGenerator did the following:

- rotated the image by no more than of +- 20%
- no horizontal or vertical shift because the hand is close to one edge in some jpgs
- normalized the 0...255 values to the float interval 0...1
- rescaled the image to 255 x 255 pixels
- returned the image itself and the one-hot encoded class based on the class directory

The next steps are:

- define the CNN model with at least one hidden convolution layer
- compile the model
- run the model and save the metrics
- analyze the results (loss, accuracy)