Thibaut Baguette and Mohamed Elsamadouny
MAIS 202 Bootcamp
16 November 2021

Deliverable 3

1. Final Training Results

The preliminary results presented in Deliverable 2 were accuracy scores of 1.0 both for the training and testing datasets. The explanation in Deliverable 2 was that the testing set might have been too small compared to the training set.

After talking about the issue between ourselves and to family and friends, we discovered a major mistake in the data preprocessing and thus, the ML model. The data gathered for the sign recognition was formatted as (x, y) coordinates representing the position on the image, normalized in the closed interval [0, 1]. The issue was then that the same sign, done on the left or right side of the video feed would give very different datapoints. Luckily, this was easily fixed; the hand identification results from mediapipe are also available as "world coordinates," where they are instead measured in meters, *with respect to "the hand's approximate geometric center"*[1]. Using this data instead means that the position of the hand with respect to the camera has no impact on the gathered data. Fixing this issue drastically improved the model's performance, especially with the hand farther away from the camera.

Concerning the training and testing datasets, the Kaggle dataset's premade testing data (one image per category) was ignored, and a train/test split was done with the training data. The split was 2700/300, or 90/10; this was achieved with `sklearn.model_selection.train_test_split()`. With this new split, and the change in preprocessing described above, the new training and testing accuracies were of 0.8992 and 0.9017, respectively (see Table 1).

Table 1. Preliminary and final results of the sign recognition model.

Data	Using "absolute" coordinates (preliminary)		Using "relative" coordinates (final)	
	Training data	Testing data	Training data	Testing data
Dataset size	18 000	6	16200	1800
Accuracy	1.0	1.0	0.8992	0.9017

[1] Hands - mediapipe (google.github.io)

The final results appear to be much realistic. In addition, it does not seem that the data is overfit, since the training accuracy is not much higher than the testing accuracy (in this case, it is even lower). Table 2 shows the confusion matrix of the testing results. One can notice that most cells off the "main diagonal" are filled with zeros or ones, except from the Move column. This indicates that quite many images were wrongly predicted as 'move.' The model therefore has a tendency to "overpredict" the move category.

Table 2. Confusion matrix of the sign recognition model test data

Actual \ Predicted	Line	Ellipse	Rectangle	Triangle	Move	Delete
Line	263	1	0	0	32	0
Ellipse	0	286	0	0	35	0
Rectangle	0	0	243	0	50	0
Triangle	0	0	0	273	25	0
Move	0	0	1	0	302	0
Delete	0	0	0	0	33	256

2. Final demonstration proposal:

For our final demonstration, we will integrate our model into a webapp using Flask. The webapp will consist of: a camera feed where the user will be able to see themselves and what our model is detecting, a canvas where the shapes are drawn and can be moved/deleted and a legend of hand signs that the user can refer to while making the hand signs.

For the camera input, we will be following this tutorial **Video Streaming Using Webcam In Flask Web Framework** which uses OpenCV to render the camera input frame by frame and then display it in the webapp using Flask.

For the canvas and drawing of the shapes, we are using pygame which runs as a window on our local computer. We are then planning on projecting that window onto the webapp, perhaps as another image feed using the same tutorial as above.

Including a legend of hand signs will be done using Flask.