**Hand gesture recognition project (ASL)**

Dataset:

1. <https://www.kaggle.com/grassknoted/asl-alphabet>

**Chapter 1: Problem statement & Motivation**

The sign language is a visual language used by the people with speech and hearing disabilities to communicate in their daily lives. One of them is ASL, which is the most widely used sign language in the world. It provides a set of 26 gesture signs to represent the letters, and a set of 10 numeric gestures, from ‘0’ to ‘9’.

A good hand recognition system could let people connect more readily, even if they don't speak the same language, by interpreting sign language in real time and without the need for prior knowledge.

**Chapter 2: Literature review**

The sign language is a visual language used by the people with speech and hearing disabilities to communicate in their daily lives. One of them is ASL, which is the most widely used sign language in the world. It provides a set of 26 gesture signs to represent the letters, and a set of 10 numeric gestures, from ‘0’ to ‘9’.

A good hand recognition system could let people connect more readily, even if they don't speak the same language, by interpreting sign language in real time and without the need for prior knowledge

**Chapter 3: Description of the proposed approach**

Once I have loaded the train and test images, along with their labels, using sklearn.model\_selection.train\_test\_split I have split the data into 2 sets:

A training set

A test set.

Before feeding it to the model, I have had to preprocess the data. Firstly, I have started with the labels. Their encoding was: A→0, B→1, C→2, …. Z→25, ‘nothing’→26, ‘space’→27. Basically, each label was assigned an integer.

Afterwards, I have one-hot encoded the labels, which means that each label will be associated with a vector made up of 0s and 1s, with the size of 28. This vector has 1 only on the index associated with the label, and 0 elsewhere. For example, the one-hot encoding for M={1,2,3} would be:

1 = [1, 0, 0]

2 = [0, 1, 0]

3 = [0, 0, 1]

An image has 3 channels for red, green, and blue, whose component values can range from 0 to 255. To get a smaller interval, i have divided the numbers by 255, obtaining an interval of [0, 1].

Now that we have our data hot and ready, we have to define a convolutional neural network(CNN), which takes as input an image of a sign language, and outputs the probability of the said image belonging in our predefined categories (A-Z, space, nothing).

I have used a sequential model, which is a linear stack of layers, and I have started adding layers to it. Conv2D is the first layer, and because of that i have had to specify the input shape ( 64,64, 3), as our images are RGB, 64 by 64, with 3 channels. The MaxPooling downsamples the input along its spatial dimensions (height and width). The Dropout layer helps in preventing overfitting. Flatten will literally flatten the input into a one dimension vector. Dense is our last layer, with 28 units, each corresponding to A-Z, space, nothing.

The metrics used are accuracy and recall, and after compiling and fitting the data, after 2 epochs, I have obtained the following metrics:

Epoch 2/2

Epoch 3/5 756/756 [==============================] - 1967s 3s/step - loss: 0.0777 - accuracy: 0.9735 - recall\_6: 0.9703

Accuracy for test images: 98.0 %

Accuracy for evaluation images: 100.0 %

I have obtained a better accuracy compared to the squeezenet model (87 vs 92).

I have tried various models until I have reached this one. I have started with a simple model and I have obtained a 82% accuracy:

Epoch 2/2 756/756 [==============================] - 166s 219ms/step - loss: 0.5776 - accuracy: 0.8245 - recall\_2: 0.7287

Climbed to a 92% accuracy when I have stacked two convolutional layers one after another:

756/756 [==============================] - 1520s 2s/step - loss: 0.2460 - accuracy: 0.9215 - recall\_3: 0.8988

Added another stack of two layers with maxpooling & dropout, and the accuracy climbed by 2%:

Epoch 2/2 756/756 [==============================] - 1846s 2s/step - loss: 0.1736 - accuracy: 0.9411 - recall\_4: 0.9282

After 3 epochs, I have reached a 97% accuracy:

Epoch 3/5 756/756 [==============================] - 1967s 3s/step - loss: 0.0777 - accuracy: 0.9735 - recall\_6: 0.9703

**Chapter 4: Presentation**

Using python and cv2, we process each frame and we analyze the region of interest within a square, frame which we pass through our model to get our prediction. The letter is displayed on the left side of the screen.

**Chapter 5: Conclusion**

The purpose of this project was to achieve & work with a model that could accurately predict sign language over video. With an accuracy of 96%, I believe I have achieved my purpose.

[1]<https://www.researchgate.net/publication/271556538_Low_cost_approach_for_real_time_sign_language_recognition>

[2] <https://arxiv.org/ftp/arxiv/papers/1905/1905.05487.pdf>