Anvesh Kumar Voona
Email: avoona@asu.edu
Course: CSE 535: Mobile Computing
Instructor: Dr. Ayan Banerjee

Final Exam
ASU ID: 1219792743
Term: Spring 2021
Due Date: 2^{nd} May, 2021

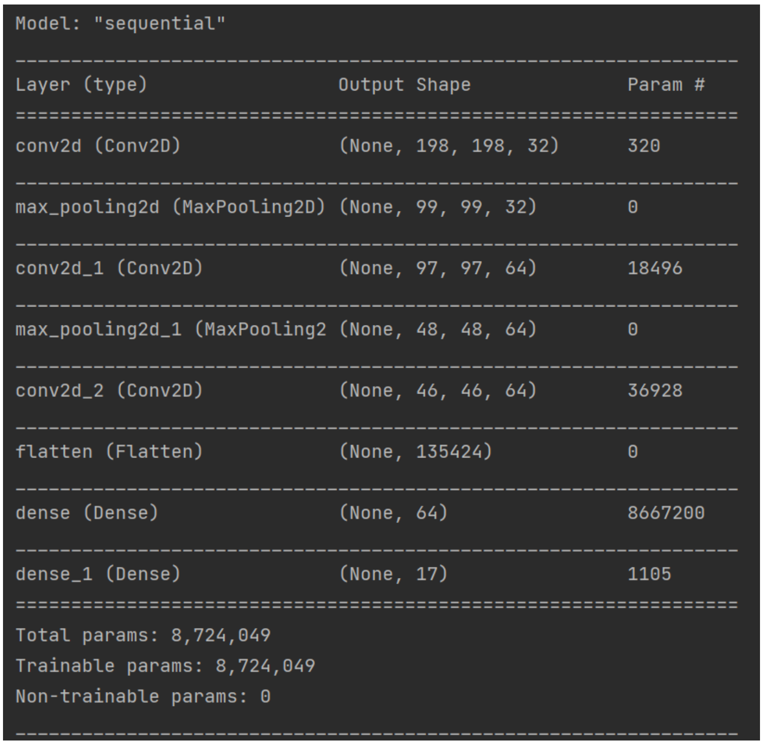
Solution
In the part 2 of the assignment, we concentrate on the gesture recognition part, Where the given testing gestures are matched against the actual gestures used in the training process. Here are the tasks performed in the "main.py":

- Generating the penultimate layer for the video files in the "test" and the "traindata" folders.
  1. Look for the ".mp4" files in the directory and make a list of all videos
  2. For each video extract the middle frame using the OpenCV VideoCapture function.
  3. Once the frame is captured, I use the instance of the CNN model from the hand shape feature extractor python class and extract the feature vector.
  4. Make a list of all the feature vectors in an array, So there will be two arrays one for "test" and other for "traindata"
- Calculating the cosine similarity for the feature vectors of "test" and the "traindata"
  1. There are 51 videos in the test data so there will be 51 feature vectors corresponding to each frame. Loop through every entry in the feature vectors in the "test" data set and calculate the cosine distance for each of the entries in the "traindata".
  2. Once we get the cosine distance for every entry in the train data, we take the mist similar one, Which is basically the minimum distance vector and store the index for the same. This will be an array of matched labels.
  3. Finally, we will have an array of 51 labels, The results are then printed to the "Results.csv" file in a single column.

Observations: After running multiple iterations of the code, I have observed that the model performance is very less as this is trained for alphabet gestures and the feature vector is also having 27 labels. So I trained the mode with the gestures in the training data and the expert gestures with 17 labels as the output of the model. The model creation process is described below

Generating Training Data: To create a new model and train, the first step is to generate the training data. So in the generate python file generates the training data from the folder passed in the frame extract method. During the generation of training data, The frames at the center of the video is used and the frames around the center are taken. To make sure that the hand gestures used by the left hand are also recorded a mirror image is also taken as a sample.

Model Training: In the training process we create a model with 3 Convolution layers, 2 Max Pooling layers, One Flatten layer and Two Dense layers. Then the model summary looks a below.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 198, 198, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 99, 99, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 97, 97, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 64)        36928
_________________________________________________________________
flatten (Flatten)            (None, 135424)            0
_________________________________________________________________
dense (Dense)                (None, 64)                8667200
_________________________________________________________________
dense_1 (Dense)              (None, 17)                1105
=================================================================
Total params: 8,724,049
Trainable params: 8,724,049
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Model Summary

Then using the ”train_test_split” function of the ”sklearn” library the split and shuffle of tain and test data is done. Then the training and the validation data is passed to the model.fit function of the CNN model. Once the training is done it is saved as an .h5 file. This model is used to run the cosine similarity discussed in the main.py file