Group-15 Assignment 2: Mobile Computing

Tejaswi Paruchuri (1213268054), Thanuja Kallamadi (1217132202), Kusuma Kumari Vanteru (1217031218), Snehitha Rednam (1217164507)

Features Extraction and Models Generation:

The main aim of this project is to detect the gesture from the given key points in a json file for each gesture. For this, a model must be trained which can predict the gesture based on the given input of a gesture's json file which has key points. We have trained models using the concept of Supervised learning which mean we will have features with labels. Using these features and labels we must generate a model which will predict the label of the features given as input later.

Dataset:

The dataset consists of the videos which were recorded thrice for each gesture during project 1. From those 20 gestures only 6 gestures (fun, really, hope, communicate, mother, buy) were chosen as a part of this project. Since we have 4 members in our team, we have a total of 72 videos (4*6*3) in total. For these 72 videos first images were generated (30 images approx. for 1 second length of video) and from these images key points were generated using the instruction steps and code provided in https://github.com/prashanthnetizen/posenet_nodejs_setup repository. Now we have a data set with 72 json files along with the json files provided as a part of this project to complement the training. Each json file will have approximately 150 features (30*5) since each video is of length 5 seconds and for each second, we will have 30 images, as a result we will have one feature per image in the json file.

Feature Extraction:

The main part of this project depends on how effectively features were extracted from the given dataset of json to train a model. As in most of the gestures, the movement is based on wrists and elbows. For each feature in a json file we will consider key points for nose, left ear, right ear, left eye, right eye, left shoulder, right shoulder, left elbow, right elbow, left wrist and right wrist to extract features. The features are extracted by taking norm using "linalg.norm" of numpy package between left wrist and each of the remaining key points mentioned above.

Norm(nose, left wrist) , Norm (left ear, left wrist) , Norm (right ear, left wrist), Norm(left eye, left wrist), Norm(right eye, left wrist), Norm (left shoulder, left wrist), Norm (right shoulder, left wrist), Norm (left elbow, left wrist), Norm(right elbow, left wrist) and Norm(left wrist, right wrist)

Similarly, between right wrist and other key points, left elbow and other key points and right elbow and other key points.

So, from the above for each feature in a single json file we will have 38 values and at the end we will append label (which is the name of the gesture). These 39 values will be appended as a row in csv file which will be used to train a model later. Each json (each gesture) will have approximately 150 entries in the csv file.

These extracted features are common for all the 4 Machine learning model we have used to train for this project.

Models Generation:

Entire data set we got in the csv was split into training and testing data in the ratio of 70:30 with a random state of 7 for KNN and Decision Tree, 40 for Neural Network and Random Forest using **test_train_split** of model selection of sklearn package. Before that features were preprocessed by using **MinMaxScaler** so that data set will be transformed in the range of 0 to 1. The data that was split for training is used to train the machine learning models whereas the testing data is to validate the trained models.

Below are the 4 models which we used for training:

1) **K Nearest Neighbors:**
   In KNN entire data set is represented as a model which means it stores entire training data set that is used while training the model. Predictions are made by matching the new feature with the K nearest neighbors and predicts the label by using training dataset directly.
   Parameters chosen:
   Number of neighbors(n_neighbors): 99
   Metric: Euclidean

Model was trained by varying both the number of neighbors and metrics with various values. The above parameter values were chosen as they have provided us the model with highest accuracy.

2) **Decision Tree:**

Decision Tree follows a tree like structure in which leaf nodes represent labels and non-leaf nodes represent features. So, based on this approach a tree like structure will be formed from the training data set during model training. From this tree structure predictions will be done by mapping features to the non-leaf nodes and reaches a leaf node to get the label.

Parameters chosen: Default

As the model's accuracy wasn't improved by changing the parameters, we have chosen the default ones.

3) **Random Forest:**

Random forest which is also called Random decision forest is a collection of multiple decision trees from the given training set of data. So as similar to decision tree model it will construct tree with labels as root nodes but here, we will have multiple such trees. While predicting the label for the given feature either the mode or mean of the labels which was predicted from multiple trees is taken as the final output label

Parameters chosen: Default

As the model's accuracy wasn't improved by changing the parameters, we have chosen the default ones.

4) **Neural Networks (using Multi-layer Perceptron classifier)**

Multi-layer Perceptron classifier trains mainly using Back Propagation techniques where some form of gradient descent and gradients are calculated using backpropagation. In this classification technique it minimizes the Cross- Entropy loss function giving probability estimates per each sample.

Parameters chosen:

hidden_layer_sizes: 30 number of neurons were chosen to be at the hidden layer

max_iter: 10000 number of iterations were chosen if the solver doesn't converge.

solver: adam was chosen as the stochastic solver to train this model

random_state: 21 seed size is chosen which is used by the random number generator.

tol: 0.000000001 is chosen as the tolerance for the optimization

These trained models were saved into pkl files which can later be used for predicting other gestures.

Accuracies of testing data during training:

K-Nearest Neighbors: 0.9198
Decision Tree: 0.9688
Random Forest: 0.9874
Neural Networks: 0.9037
Average Accuracy: 0.9449
Model with highest Accuracy: 0.9874

Accuracies of testing data on custom data:

K-Nearest Neighbors: 0.6111
Decision Tree: 0.6111
Random Forest: 0.7222
Neural Networks: 0.9444
Average Accuracy: 0.7222
Model with highest Accuracy: 0.9444

Execution Steps:

The main code for extracting features and training models is in FeatureExtraction_ModelGeneration.py file

1. Make sure that at line 22 in FeatureExtraction_ModelGeneration.py file json path variable has the correct path which has json files to extract features.
2. Using the command line arguments while executing the python files values of the variables choice and fileName can be changed.
    - Options for choice variable are 1- only to extract features, 2- only to train models and 3- both for feature extraction and model training. By default, the choice is 3
    - Variable fileName can be used to give the fileName of the csv which will be generated with feature extraction or in case of model training the csv file which can be used for training models. By default, the fileName is feature_points.csv

```
E:\Courses\MC\DownloadFilefromURL\assignments\2\Submission_final>python2 FeatureExtraction_Mode
usage: FeatureExtraction_ModelGeneration.py [-h] [-choice 3]
                                            [-fileName feature_points.csv]

optional arguments:
  -h, --help            show this help message and exit
  -choice 3             1 - for feature extraction 2 - for training model 3 - for both 1 & 2
  -fileName feature_points.csv
                        enter file name to save features

E:\Courses\MC\DownloadFilefromURL\assignments\2\Submission_final>python2 FeatureExtraction_Mode
fp1.csv
```

Whereas, file predict_gestures.py is used to predict the gestures of the given json files and calculate the accuracy on how correctly models have predicted the gestures. To execute this, make sure at line 42 predict_gesture.py file json path variable is pointing to the correct path which has json files.

Hosted Service URL and Result:

URL:

https://mcspring2020-group15.wm.r.appspot.com/predictgesture

Json Result:

```
{
    "1": "gesture predicted by KNN",
    "2": "gesture predicted by
DecisionTree",
    "3": "gesture predicted by Random
Forest",
    "4": "gesture predicted by Neural
Networks"
}
```

Sample Output:

```
1  {
2      "1": "mother",
3      "2": "hope",
4      "3": "mother",
5      "4": "mother"
6  }
```

Conclusion:

Models were trained using the gesture videos and are tested to predict the gestures and were successfully deployed on Google Cloud Platform.