UPPSALA
UNIVERSITET

Project Report

Smart Gesture Recognition System

Group 7

Anil Poudel Hosam Mohammed Praveen Swamy
Uvais Karni

June 1, 2020

Project Introduction

The smart hand gesture recognition system can detect six different hand gestures, as listed in Table 1, and reacts accordingly. The system consists of three subsystems. The main building blocks of these subsystems are TensorFlow, machine learning techniques and virtual Furhat robot for reactions. The input of the system is a video containing a hand gesture, first subsystem processes the video and extract landmarks of the hand in each frame then pass it to the trained machine learning model to decide which hand gesture is present in each frame then the detected gesture will be shown to the user in order to give a voice command to Furhat robot with the detected gesture to do facial expressions and speech sentences depending upon the type of gesture.

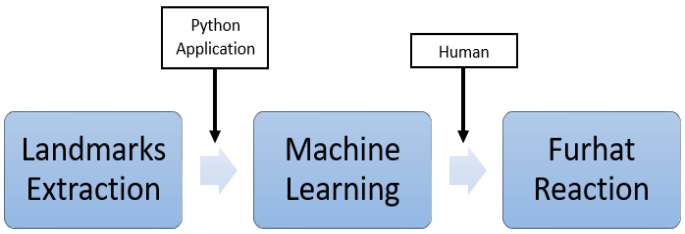

Figure 1: System Design.

Description of the system

The project is developed in three phases and the subsystems constituting the final outcome are described in the below paragraphs.

Landmark extraction (First Sub-system)

- Design

  This subsystem utilizes a pre-trained model from [1] to detect the keypoints on different hand gesture images. The detector from [1] follows the architecture of Convolutional Pose Machines(CPMs).

  The subsystem 2 which is trained using the provided CSV data-set requires the landmark points to be structured as dorsal and palm points separately. To achieve this a separate SVM classifier is trained on a limited set of frames for each video to distinguish between the input frames as palm and dorsal regions. Then each video is read from the given data-set and the region type is predicted using the trained classifier. After that the pre-trained model[1] is loaded for keypoints detection and the landmark points are drawn. Based on the region type determined from the classifier the landmark points are structured to be passed to the subsystem 2 model to predict the final gesture type.

  To connect the subsystem 1 with subsystem 2 the model which was trained in the second subsystem using the CSV data-set is loaded and the gesture type is predicted by passing these landmark points obtained as input.

- Materials

  The trained model and the weights file from [2] is used for landmarks extraction which is an open source project for Hand pose hosted on GitHub.

  Tensorflow is used for training the models in both subsystems. SVM classifier is used for region classification i.e. palm and dorsal. The output video files with all the landmark points and the gesture type are saved as MP4 files. Sample result is shown in Figure 3 below.

- Procedure

  CPMs predict a confidence map for each keypoint representing the keypoint's location as a Gaussian centred at the true position [1]. The predicted confidence map corresponds to the size of the input image patch and the final position of each keypoint is obtained by finding the maximum peak in each confidence map.

  The detector is trained using the convolution stages of a pre-initialized VGG-19 network [1].The feature extractor comprises of conv4_4 with two additional convolutions producing 128-channel features F [1].

  After getting the feature maps, a prediction stage produces a set of P confidence or score maps. One confidence map for each keypoint p. Each stage of prediction takes the score maps as input from the previous stage and is concatenated with the image features F. This produces P new score maps S', one for each keypoint. These maps are then resized to the original input patch size and extract each keypoint location as the pixel with the maximum confidence in its respective map.

- Results

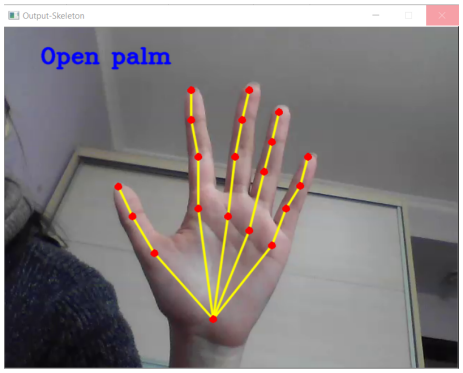

Figure 2: Open palm output.

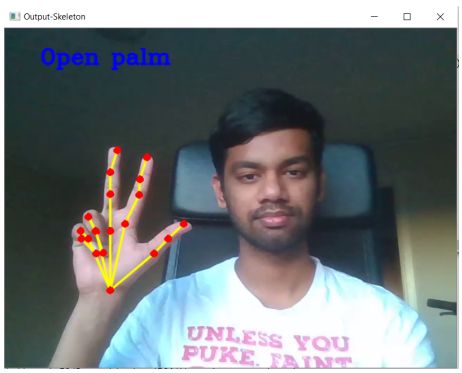

Figure 3: Three fingers palm output.

- Discussion
  - First Subsystem resulted into some errors while classifying the gestures apart from open palm and dorsal palm. Since the second subsystem was trained with CSV data-set with landmark points structured according to the gesture types e.g. the three fingers dorsal gesture contains landmark points for both palm and dorsal. To frame the landmark points detected in the first subsystem the gesture type needs to be known beforehand.

  - The pre-trained model marked the landmark points even for the hidden regions for e.g. in fist palm only dorsal points needs to be detected but the model detected palm points as well.

Gesture recognition (Second Sub-system)

- Design

  Second Subsystem is developed using Keras, a deep learning API where the model is designed with three dense layers and relu activation function in the hidden layers and sigmoid function in the output layer. Stochastic gradient descent(SGD) is used as an optimizer function and categorical cross entropy as loss function. The model is trained for 10 epochs.

- Materials

  Tensorflow is used as backend library and Keras API is used to build the model and training. Jupyter notebook is used for programming.

- Procedure

  The csv dataset is loaded and the data is normalized using min-max scaler. The gesture labels are converted into binary values using LabelBinarizer(). The model is built using 3 dense layers- 1 input, 1 hidden and 1 output layer. The dataset is split into training and testing sets with 80:20 ratio. The model is trained on training set for 10 epochs and the model is saved to be used in subsystem 1 for predicting the gestures.

- Results The model resulted into 100% accuracy and a value loss of 0.0052.

Responsive behavior (Third Sub-system)

The main functionality in the third subsystem is to react appropriately by Furhat social robot to the detected hand gesture that is detected by first and second subsystems.

- Design

  The initial design was to connect first and second subsystems to Furhat robot via TCP connection. Due to time limit, this feature is replaced by involving a human in the communication by reading the output of the second subsystem and give voice command to Furhat robot in order to react.

  Furhat robot reactions consists of two aspects: Facial expressions and saying a sentence. The input and output of this subsystem is shown in Table 1.

- Implementation

  The virtual robot is implemented using Furhat SDK and Kotlin scripting language. The robot waits for the user to attend in it's detection zone, then it pays attention to the user and asks for showing hand gesture. The user can say one of the options listed in the Table 1, for each hand gesture the user has the option to say the gesture directly or say the corresponding number of the gesture. For example, saying "Open Palm" or "One" results

in the same reaction, the robot will say "Hello" and nod. If the received voice command is not one of the known options, the robot responds to the user that it could not understand the command and asks for a repetition.

Input	Motion	Speech
Open palm/One	nod	hello
Open dorsal/Two	anger	you can't see me
Fist palm/Three	shake	Rebellion never goes without consequences.
Fist dorsal/Four	open eyes	Ready to fight
Three fingers palm/Five	big smile	Hurry up, let's party
Three fingers dorsal/Six	disgusting	Sorry that's rude

Table 1: System responses according to the detected gesture.

End to End Subsystem (Specialisation)

- Design

  An Object Detection API to train a Fast Region Convolution Neural Network is used. The reason we chose Fast R-CNN is that it can classify multiple objects with low latency. The input features to the network are raw frames from the videos and output is a CSV file containing the desired class (Hand Gesture) and position of the hand in the frame. The Faster R-CNN has three components:
  - Convolution Layers: To extract features from images.
  - Region Proposal Network: To check whether the desired object is present or not. If the object is present then a bounding box for those object are given out.
  - Classes Detection: Here the bounded region part of the images are classified. It simply flattens the bounding region of the image and passes it through several dense layers and then a final Softmax layer which returns probabilities for each class.

- Requirements

  Tensorflow is used as backend library and Object Detection API is used to load the model and training. Command prompt and Python Idle IDE are used for programming and excecution.

- Implementation

  From video data set provided, frames at interval of 50 were extracted from all the video and this data was split in test and train in the ratio of 20:80 which resulted in having 164 frames for train and 34 for test. These frames are input features and out for network was generated manual for almost 200 frames using tool called Labellmg which generates xml file containing label and bounding for each but the issue is that we cannot

directly provide this 200 xml as output to network. So all the details were append into separate CSV files for train and test. Now to reduce the size of the data and increase performance of training the data are saved as binary record files. While saving the frames and labels as record file, the labels are encoded from categorical text to categorical numerical format. The record file are used to train the Fast R-CNN.

- Results

  It took around 11 hours and 5400 steps later to reduce the error to a respectable quantity of 0.07 as the API did not support my GPU.

Conclusion

Over a short period, we have successfully worked together to create a simple smart gesture recognition system.We were able to complete three subsystems. In the first subsystem, we worked with landmark extraction. In the second subsystem, we successfully worked with gesture recognition and in the third subsystem, we used Furhat robots which responds to the voice commands given by the users. Along with the three subsystems, the specialization part - End-to-end learning was successfully implemented.

References

[1] Hand Keypoint Detection in Single Images using Multiview Bootstrapping,Tomas Simon, Hanbyul Joo, Iain Matthews, Yaser Sheikh *arXiv:1704.07809*

[2] Notes, https://github.com/spmallick/learnopencv/tree/master/HandPose

[3] Furhat,https://docs.furhat.io/gen1/tutorials/tutorial_presentation/

[4] Object detection API, https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10#6-run-the-training

[5] Fast Region Convolution Neural Network, https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html

Appendices

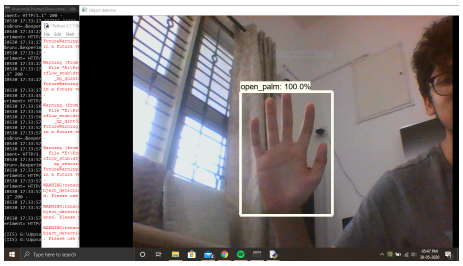

Figure 4: open palm

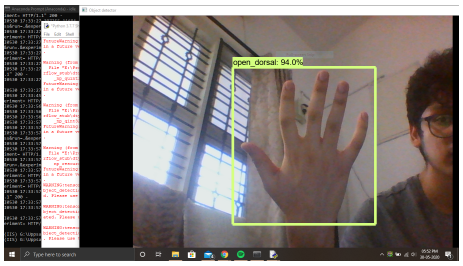

Figure 5: open dorsal

Figure 6: fist palm

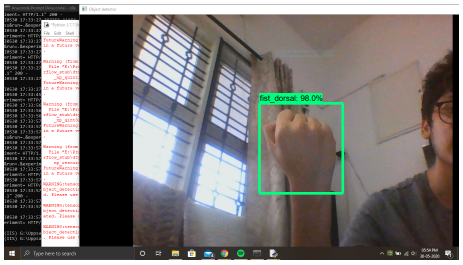

Figure 7: fist dorsal

Figure 8: three finger palm

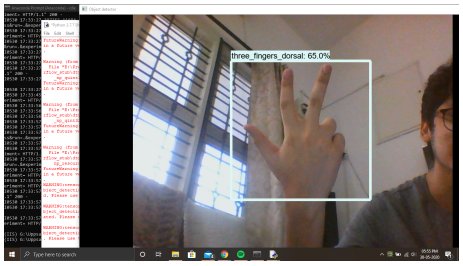

Figure 9: three finger dorsal

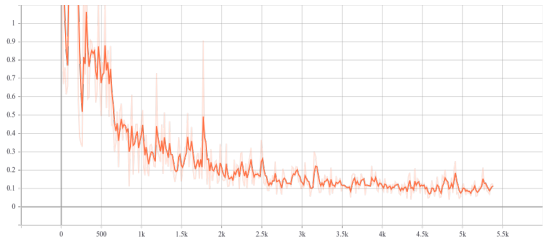

Figure 10: total loss