ECE 6372
Advanced Hardware Design

Interpreting American Sign Language (ASL) for Bot control using Neural Networks

Team Mira:

- Manojna Sistla
- Revathy Venkatesan
- Srinidhi Parshi
- Naga Nikitha Kakani

1. Introduction

The American Sign Language system is the most prolific system of communication used by the differently-abled people. In this project, we developed a system that controls the motion of the robot as per the ASL commands. Neural networks are trained using the existing data set of ASL and the trained network is used in identifying the sign shown by the user. The information of the interpreted sign by the network is sent to the bot using the Bluetooth module. Upon receiving the information from the Bluetooth module, the robot moves as per the commands.

American Sign Language (ASL):

American Sign Language (ASL) is a visual language, the primary language of people who are deaf and hard of hearing. With the signing, the brain processes linguistic information through the eyes. The shape, placement, and movement of the hands, as well as facial expressions and body movements, all play important parts in conveying information. Sign language is not a universal language — each country has its own sign language, and regions have dialects, much like the many languages are spoken all over the world. Like any spoken language, ASL is a language with its own unique rules of grammar and syntax. ASL is used predominantly in the United States and in many parts of Canada.

Neural Networks:

Neural networks are a set of algorithms that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

- Training a Neural Network means making a neural network learn a new capability from existing data.
- The inference is defined as applying the learned capability to new data.

2. Hardware Components

- Arduino Uno
- L293D Motor Driver IC
- Raspberry Pi
- HC-05 Bluetooth Module
- Neural Compute Stick

2.1 Arduino UNO:

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller. The board is equipped with sets of digital and analog input/output pins that may be interfaced with other several boards and circuits. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button.

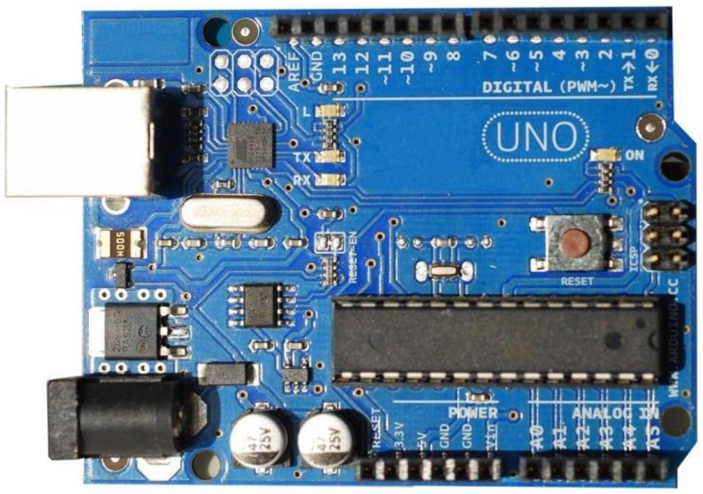

Fig: 2.1 Arduino UNO

2.2 L293D Motor Driver IC:

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive in either direction. L293D is a 16-pin IC that can control a set of two DC motors simultaneously in any direction. It works on the concept of H-bridge. H-bridge is a circuit that allows the voltage to be flown in either direction. In a single L293D chip there are two h-Bridge circuits inside the IC which can rotate two dc motors independently.

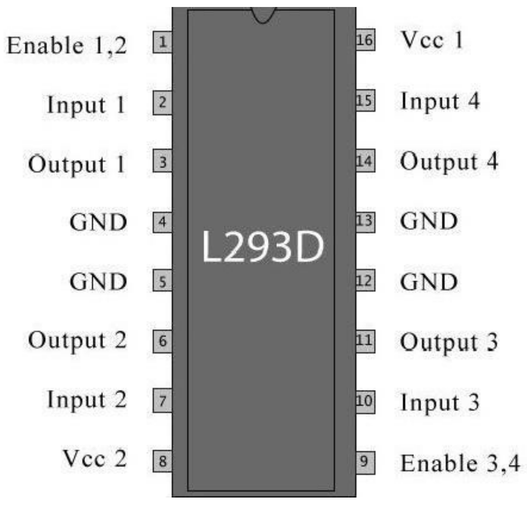

Fig: 2.2 Motor Driver IC

The robot consisted of 4 wheels connected to DC motors that ran with the help of L293D motor driver. The HC-05 Bluetooth module was connected to the TX and RX pins of the Arduino. This ensured that the Arduino received the commands that were sent via Bluetooth.

2.3 HC-05 Bluetooth Module:

HC-05 is a Bluetooth device used for wireless communication with microcontrollers using serial communication (USART). The default settings of the HC-05 Bluetooth module can be changed using certain AT commands. As the HC-05 Bluetooth module has a 3.3 V level for RX/TX and the microcontroller can detect 3.3 V level, there is no need to shift the TX voltage level of the HC-05 module. But we need to shift the transmit voltage level from the microcontroller to RX of the HC-05 module.

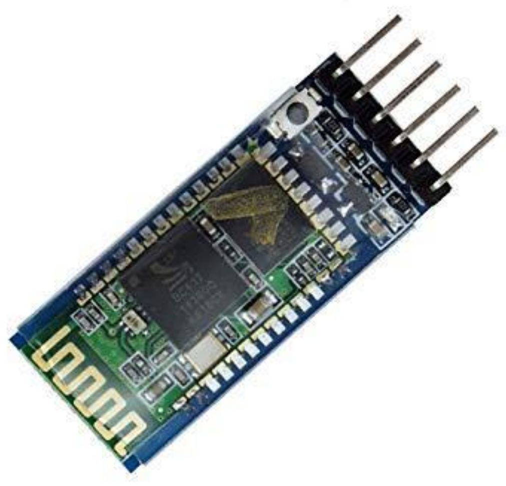

Fig: 2.3 HC-05 Bluetooth Module

2.4 Raspberry Pi:

Raspberry Pi is a small single board computer. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer. Raspberry Pi is popularly used for real time Image/Video Processing, IoT based applications and Robotics applications.

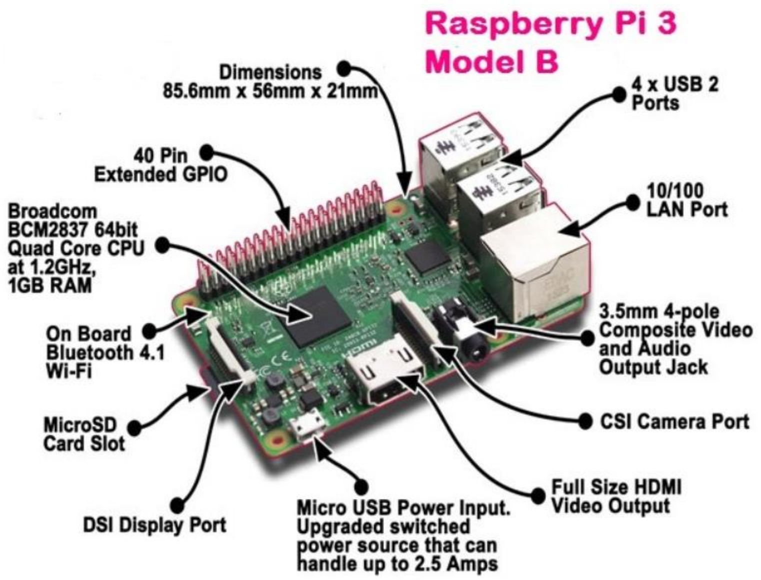

Fig: 2.4 Raspberry Pi Module

2.5 Neural Compute Stick:

Intel’s Neural Compute stick is a low-power, tiny, fan-less device that can be used to develop, fine-tune and deploy deep learning neural networks(DNNs) on edge devices that operate on power constrained environments. The version of NCS that we are using consists of Intel® Movidius™ Myriad™ 2 vision processing unit (VPU), with 4Gbits of LPDDR3 DRAM, imaging and vision accelerators, and an array of 12 VLIW vector processors called SHAVE processors. It is compatible with operating systems x86-

64 with Ubuntu(64-bit) 16.04, and Raspberry Pi 3 with Raspbian stretch, and can be prototyped with devices such as ARM cores and Raspberry-pi

3 IMPLEMENTATION

3.5 Model of Bot:

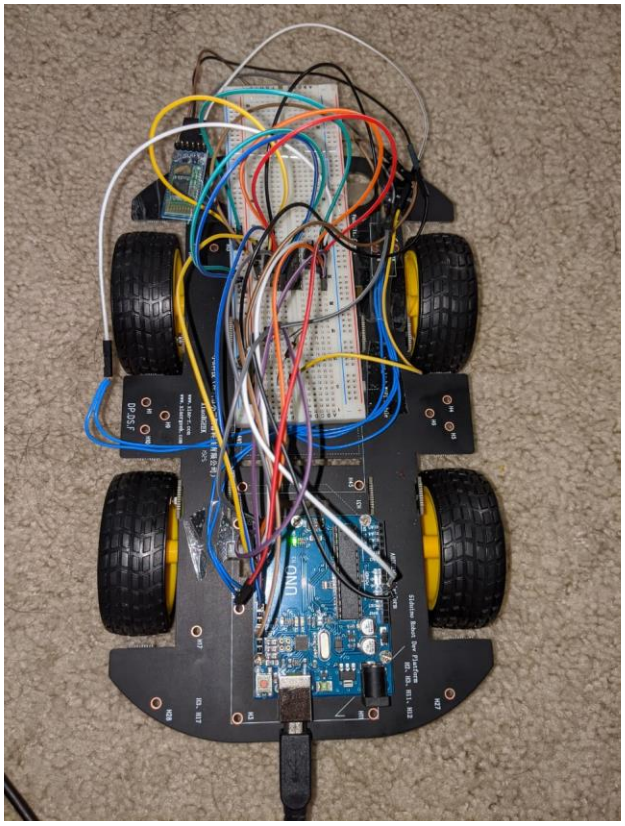

Fig 3.1 Constructed Bot

PIN Connections:

- VCC of Bluetooth connected to VCC of Arduino
- GND of Bluetooth connected to GND of Arduino
- TX of Bluetooth connected to RX of Arduino
- RX of Bluetooth connected to TX of Arduino

PIN Connections to L293D:

- VCC1 and VCC 2 connected to VCC of Arduino
- 4 GND of L293D connected to GND of Arduino
- The 4 DC motors are connected to the 4 inputs of the L293D, right two motor wires to input 3 & 4, left two motor wires to input 1 & 2. Outputs 1,2,3&4 of L293D are connected to Arduino digital pins 4,5,6&7 respectively.
- Enable pins 1,2 & 3,4 are connected to digital pins 10, and 11 of Arduino.

3.6 Training and Inference of Neural Network

Framework

- **TensorFlow**: TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Model

- **Inception v3:** Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years.
- The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs.

Libraries

- **OpenCV:** (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. We mainly used this library to analyze the video captured and image processing for detecting ASL gestures. OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:
- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.
- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.

Matplotlib: Matplotlib is the most popular data visualization library in Python. It allows us to create figures and plots and makes it very easy to produce static raster or vector files without the need for any GUIs. We used matplotlib for getting the visual box for capturing the gesture.

NumPy: NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. We used to numPy to process the array of images using fourier transformation to get the exact pixels of images.

3.2.1 Training and Inference on Linux:

Prerequisite:

- This project is developed using Python 3.7. For accurate results, it is recommended to use python 3.7. Python 3.7 can be installed using: **apt-get install python3.7**
- Install pip3 using: **apt-get install python3-pip**
- Install TensorFlow version 1.2.1 using pip3: **pip3 install tensorflow==1.2.1**
- Install matplotlib version 2.0.2 using pip3 : **pip3 install matplotlib==2.0.2**
- Install numPy version using pip3: **pip3 install numpy==1.13.0**

a. **Follow the following steps for installing OpenCV**

   Install the following dependencies:

   - sudo apt install build-essential cmake git pkg-config libgtk-3-dev
   - sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev
   - sudo apt install libjpeg-dev libpng-dev libtiff-dev gfortran openexr libatlas-base-dev
   - sudo apt install python3-dev python3-numpy libtbb2 libtbb-dev libdc1394-22-dev

b. Create a directory which will hold the repositories and clone the OpenCV's and OpenCV contrib repositories with the following commands:

   - mkdir ~/opencv_build && cd ~/opencv_build
   - git clone https://github.com/opencv/opencv.git
   - git clone https://github.com/opencv/opencv_contrib.git

c. Configuring OpenCV with CMake

   - cd ~/opencv_build/opencv
   - mkdir build && cd build
   - cmake -D CMAKE_BUILD_TYPE=RELEASE \
     - -D CMAKE_INSTALL_PREFIX=/usr/local \
     - -D INSTALL_C_EXAMPLES=ON \
     - -D INSTALL_PYTHON_EXAMPLES=ON \
     - -D OPENCV_GENERATE_PKGCONFIG=ON \
     - -D OPENCV_EXTRA_MODULES_PATH=~/opencv_build/opencv_contrib/modules \
     - -D BUILD_EXAMPLES=ON ..

d. Compiling OpenCV:

   - make -j8

e. Installing OpenCV
    - sudo make install

Overview of steps involved in running the training and inference part

step 1: copy all the code and graph to the respective folder.

step 2: copy the image that you want to use for prediction in the same folder.

step3: run the script classify.py with path to the the image that you want to use for prediction.

step 4: if you want to use webcam for live detection, then run the script classify_webcam.py

Note: For detailed ideas on training and inference, please refer to the machine learning tutorials document.

3.2.2 Inference Using Raspberry Pi:

Prerequisites:

- Connect the raspberry Pi board to a display. (For steps on how to connect Pi to display, please refer to the raspberry Pi tutorials.)
- update the raspbian OS to version 9 (stretch). (For detailed steps on how to upgrade raspbian os refer to the raspberry Pi tutorials.)
- install python3.7 on raspberry Pi. (check raspberry Pi tutorials on how to install python 3.7 on raspberry Pi)
- install pip package on raspberry pi.   sudo apt-get install python3-pip
- Download tensorflow version 1.13.1 rpm using wget (any other version of tensorflow will not work).
- wget   https://www.piwheels.org/simple/tensorflow/tensorflow-1.13.1-cp35-none-linux_armv7l.whl
- install the downloaded tensorflow rpm using pip3.
- pip3 install tensorflow-1.13.1-cp35-none-linux_armv7l.whl

Steps Involved:

- copy all the code and graph to the respective folder.
- copy the image that you want to use for prediction in the same folder.
- run the script classify.py with path to the the image that you want to use for prediction.

3.2.3 Inference on edge using Intel Movidius Neural Compute Stick:

Intel® Movidius™ Neural Compute SDK:

NCSDK comprises of two main units

- software toolkit which comprises of tools mvNCCompile, mvNCProfile, mvNCCheck that enable compilation, profiling and Validation of DNNs. Table.1 shows the description of these tools.
- Neural compute API which allows the prototyping of user application on the Neural Compute stick

Tool	Description
mvNCCompile	Compiles Neural Network model and weights into internal Movidius format that can be used by the NCAPI
mvNCProfile	Provides the layer-wise statistics, that help us evaluate the performance of our neural network on NCS
mvNCCheck	Compares the inference results obtained from the NCS with the network compilation validation.

Note: The version of NCS that we used for this project supports Caffe and TensorFlow frameworks only.

Steps taken to enable inference of ASL on NCS:

- **Obtain the checkpoint file for the network model:** In our project we have used Inception_V3 network, hence we have downloaded and extracted inception_V3 checkpoint file.
- **Train the inception_V3 on ASL Dataset:** We have used transfer learning technique to train the inception network model to detect the ASL gestures.
- **Freeze the graph for inference:** In this step, we identify all the required things like graph, weights, etc., and save them in one file. We have used freeze_graph.py in tensorflow libraries for this purpose
- **Compile the Movidius™ graph file:** Using mvNCCompile from the toolkit of NCSDK, we have converted the graph obtained in step3, into the format that the NCAPI can understand.
- **Write a program to use the Movidius graph file for inference:** We have modified the code to load the graph obtained from step4 onto NCS and perform inference

4. Challenges Faced

1. ***Environmental Constraints*****:** There are several supporting packages and libraries of python that are being used during the compilation of NCS, all these are version critical and in some of the packages such as scikit-image, the updated version no longer supports the intended purpose, hence causing errors.
2. ***Memory Constraints*****:** we have first decided to use NCS with Raspberry-pi for ASL recognition, however, in our development board we are using an 8GB SD card, which does not accommodate all the required packages.
3. ***Lack of Reference Material*****:** We have found very few supporting documents that can give us detailed idea about NCS. The version of NCS we are using is an archived version, which made it even more difficult to find the reference materials.

In the end we did successfully generate the Movidius graph file and used it for inference, but this graph generates only the predicted values of probability but does not map it to any values. Hence, we couldn’t use it for inference in our project.

5. Video Links:

- Introduction & Description: The following link consists of information about the brief description of the system.

  Introduction

- Motion of Bot in Forward and Backward direction is demonstrated in the below mentioned link.

  BotMotionFront_Back

- Motion of Bot in left and right directions is demonstrated in the below mentioned link.

  BotMotionLeft_Right

6. Future Work:

Our project can be developed with implementing more applications like equipment monitoring systems by ASL commands which makes things easy for the differently abled people. To quote an example, this project can be extended in building a robotic arm which can be used in picking or getting objects depending on the sign shown by the user. Many applications can be developed with adding some components to the project, to make the life easy and friendly equipment system for the differently abled people.