Jatin Verma, Hemant Singh, Akshay Kumar, Shubham Tripathi
Supervisor: Dr. Rajesh Kumar

A Smart Animatronic Human Face

MNIT Jaipur, India

May 2017

Jatin Verma, Hemant Singh, Akshay Kumar, Shubham Tripathi
Supervisor: Dr. Rajesh Kumar

A Smart Animatronic Human Face

Department of Electrical Engineering
Malaviya National Institute of Technology
Jaipur, India - 302017

MNIT Jaipur, India
May 2017

Certificate

This is to certify that the dissertation for the Bachelor of Technology Final year project titled, **"A Smart Animatronic Human Face"** by

1. Jatin Verma (2013UEE1223)
2. Hemant Singh (2013UEE1332)
3. Akshay Kumar (2013UEE1340)
4. Shubham Tripathi (2013UEE1770)

has been satisfactorily furnished under my supervision and shall be submitted to the Electrical Engineering Department, MNIT Jaipur towards partial fulfillment for the award of the degree of Bachelor of Technology in Electrical Engineering.

Dr. Rajesh Kumar
Associate Professor
Department of Electrical Engineering
Malaviya National Institute of Technology Jaipur
Rajasthan, India - 302017

Acknowledgements

We would like to thank all the faculty and staff who have helped us in making this report a possibility. We would like to thank our Final Year Project supervisor, Dr. Rajesh Kumar, who gave us continuous guidance, assistance, and inspiration to continue working on our project and obtain results. We would like to thank Mr. Vinod Sahai Pareek and Ms. Nikita Jhanjariya for their advice, help and aid in troubleshooting the robot.

We would also like to thank the Raman Lab Staff for helping us in repairing some of the Robot's components and allowing us to use their space for testing and programming.

Jatin Verma
Hemant Singh
Akshay Kumar
Shubham Tripathi

Contents

List of Figures

List of Tables

List of abbreviations and acronyms

DC PM	DC Permanent Magnet
Reso.	Resolution
RNN	Recurrent Neural Network
LSTM	Long-Short Term Memory Model
DMN	Dynamic Memory Network
CV	Computer Vision
HOG	Histogram of Oriented Gradients
CNN	Convolutional Neural Network
LMNN	Large Margin Nearest Neighbor
ResNet	Residual Network
FLOPS	Floating Point Operations per second
SGD	Stochastic Gradient Descent
Backprop	Back Propagation

Abstract

With the advent of automation in every industry that begun towards the end of the 20th century, the world is now seeing automation sneak into the personal lives of the people as well. From home and service automation to seeing machines takeover many tedious and banal tasks; the amalgamation of robotics and automation has made lives extremely easier with better standards of living. There has been a significant rise in the personal assistant category of robots where robots perform tasks that humans want, then and there.

A personal robot has the potential of shaping the lifestyle of a human by delivering the apt requirements of his various activities. The proposed smart animatronic human face is one such personal assistant robot that shall be able to interact with the user via its anthropomorphic interface. The human like face and features enable this proposed system to establish a better connect with the user than the otherwise dumb machines performing the same tasks. Unlike personal chatting clients like Google Allo and Siri which one can't really feel the existence of, or the other machines like printers, computers and robotic arms that are merely controlled via buttons – our proposed system has the potential of doing all such tasks while it already establishes an emotional connect and more natural interface using speech and vision.

Our proposed system is capable of simultaneously processing the camera feed as well as the voice input and perform the tasks as desired either by the gestures or voice commands. The major application of the proposed system lies in dealing with specially-abled children who suffer retarded mental growth. It has been seen that for such children a more like personal assistant can establish an emotional connect at par with the levels of a normal human. It can help such children learn by broadening their analytical and interpersonal interaction skills. We intend to make our proposed smart animatronic human face as a standalone system which can attend to such children, perform tasks as per their desires and help them learn under a human like interface without a natural one existing there.

Introduction

As robots take on an increasingly ubiquitous role in society, they must be easy for the average citizen to use and interact with. They must also appeal to persons of different age, gender, income, education and so forth. The goal is to design the human-robot interface such that untrained users can make safe and efficient use of the robot. It has been shown that education and entertainment are very promising applications for such interactive robots. For instance, robotic tour guides have appeared in a few museums and are very popular with children. Toward this goal, we thought it would be worthy to work on socially interactive robots. As the first step, an appropriate robotic platform was required for further explorations. Since such robots with the desired flexibility and sensory motor capabilities are not yet widely available, we had to develop own robot from scratch. We decided to work on our robotic face due to the prominence of the face in social exchange. Previous work on software agents supports this by showing that people would find interaction with an agent that had a human face more appealing than an agent with no face. It is even indicated that people are more willing to cooperate with agents that have human faces. Rather than working on an animation-based synthetic face, we decided to work on a physical robot face to improve its believability. There are reasons supporting this idea.

For instance, people expect that moving objects require intelligent control, while flat images likely result from the playback of a stored sequence as in film or television. Moreover, a physical robot can be viewed from different angles, it can be touched and its approach toward people may be thought of as threat. Apparently animation does not have such strong impacts on people. In addition to having a face, incorporating emotions in social robots can improve their operation for some reasons. First it helps the human-robot interaction to look more believable. More ever, it provides feedback to the user, such as indicating the robot's internal state, goals and intentions. Lastly, it can act as a control mechanism, driving behavior and reflecting how the robot is affected by, and adapts to different factors over time. Emotion expression on the face is an important biological aspect of emotion that has significant implications for how it is communicated in social situations. So, facial features are a key aspect for designing an interactive robot face. The mechanical design and implementation of the face and facial features must serve a means for controlling multiple joints and features simultaneously and exhibit different facial expressions. Robot's facial expression must be so easy to interpret that ordinary people easily recognize it.

The appearance of our robot must be such that it encourages humans to treat it as if it were a young socially aware creature. The face of the robot must reflect an amount

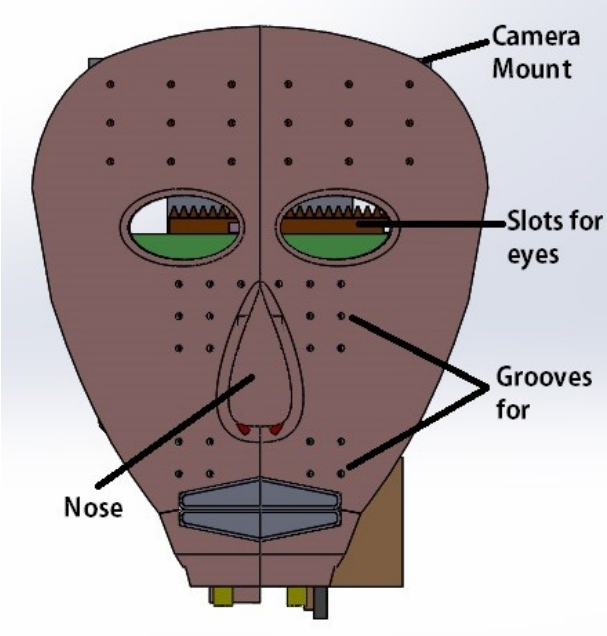

Figure 1 – Front View of the Robotic face

of robotness so that the user does not develop detrimentally false expectations of the robot's capabilities. In fact, humans have strong implicit assumptions regards the nature of human-like interactions, and they are disturbed when interacting with a system that violates these assumptions. Social robots must also possess perceptual abilities similar to humans. In particular, they need perception that is human-oriented and optimized for interacting with humans and on a human level. On the way to this goal, we should implement an anthropomorphic active vision system for our robot, capable of detecting and tracking human faces, facial features and hands. Acquired data must have high acuity for recognition tasks and for controlling precise visually guided motor movements, plus a wide field of view for search tasks, for tracking multiple objects coarsely, compensating for involuntary ego motion, etc. A simple brain must be developed for the robot so that it can display intelligent and emotional behaviors. Behaviors are realized by mapping visual stimulus and internal emotional state of the robot to its motor commands and facial expressions in a meaningful way. This can be achieved in a number of ways, e.g. heuristic rules, neural networks, state machines etc. The main objective of this project is to put all of the mentioned components together and show the possibility of developing such a complex robotic platform from scratch with very elementary and low-cost components with aid of basic tools. Multiple degrees of freedom are missing and are expected to be added in the future and in other parts of the project. The design task is to build a physical robot that encourages humans to treat it as if it were a young socially aware creature. One suggests the following characteristics for design aesthetic:

The design task is to build a physical robot that encourages humans to treat it as if it were a young socially aware creature. One suggests the following characteristics for

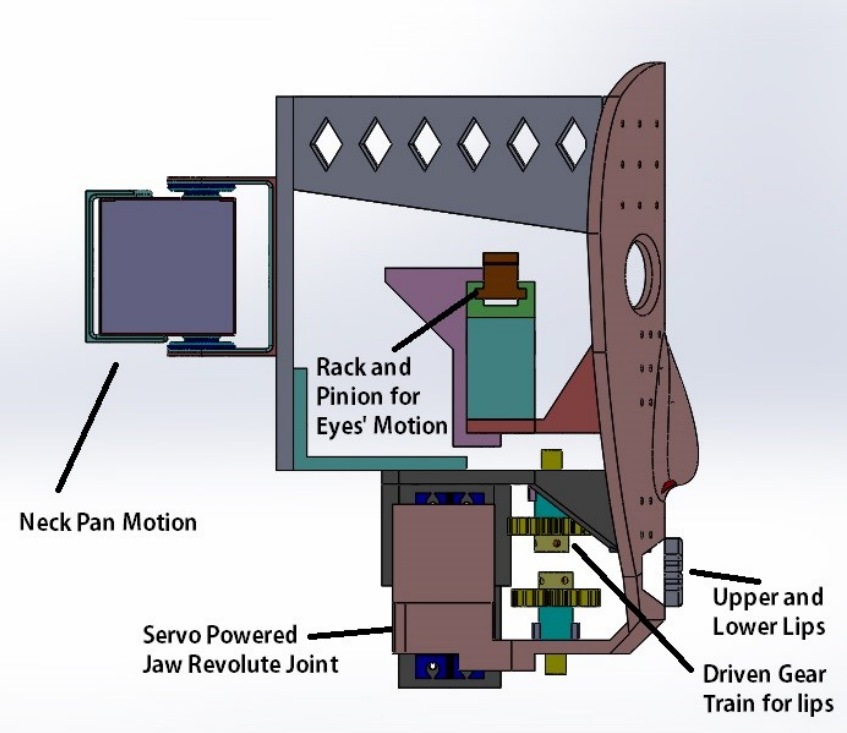

Figure 2 – Side View of the Robotic face

design aesthetic:

i. The robot should have an appealing appearance so that humans naturally fall into this mode of interaction. Physical appearance biases interaction.

ii. The robot must have a natural and intuitive interface (with respect to its inputs and outputs) so that a human can interact with it using natural communication channels. This enables the robot both read and send human-like social cues.

iii. The robot must have sufficient sensory, motor and computational resources for real-time performance during dynamic social interactions with people.

An iconographic face consisting of two eyes with eyebrows and a mouth is almost universally recognizable, and can portray the range of simple emotions useful for interaction with humans. Therefore, we focused on these facial features for development. This project report spans over various chapters where each one of them distinctively explains one function/operation of the robot. Essentially, we discuss about the tasks that we shall be achieving and the methodology used for the same. The report further illustrates the mechanical design of the robot and its operational modes – face tracking, audio-visual interaction and teleoperation. The robot can also be henceforth referred to as actroid. Figure a and 2 show the front view and side view of the CAED model of our proposed robot.

Part I

Design of the Robot

1 Components of the Design

The proposed animatronic face is supposed to resemble a normal human face in possible ways - shape, size and functional sections. In our design (facial features and neck), there are six independent motions, so called Degrees Of Freedom (DOFs). Each DOF is actuated either by a DC servomotor or a PMDC motor modified to work as a closed-loop system. Briefly, it is made of a gear train, a feedback element and a control circuit. All of these joints motor powered joints are essentially revolute joints or revolute joints with extended gear- trains/arrangements to facilitate linear actuation. The basic parts of any human face are:

1. Eyes
2. Lips
3. Eyebrows
4. Jaw
5. Cheeks
6. Forehead
7. Eyelashes
8. Nose
9. Teeth
10. Tongue
11. Ears
12. Neck

Out of these parts - **eyeballs, eyelashes, lips, jaw and tongue** are the only parts that have visible motion involved. The rest of the parts like cheeks mays seem to move - but they are merely motion of the skin and do not involve any mechanical joint. The human face essentially has only one movable mechanical joint – the jaw. It is a one DOF joint that moves in the sagittal plane. The jaw motion gives the ability to talk and express the feelings as detailed as possible. The human jaw is responsible for making certain tasks like chewing food, talking and cleaning the mouth possible. However, given the intricate design of the human face and the complexities involved in creating a complete lookalike; our project aims at replicating only a few sections of the human face. The project tries to imitate the following joints/motions of the human face.

1. Motion of the eyeballs
2. Motion of the jaw
3. Motion of the upper lips and the lower lips
4. Motion of the neck – Pan and tilt motion

Each of these DOFs has individual control and work via embedded circuitry in real time to achieve the different motions. Following sections discuss the individual designs of all the parts of the proposed human face.

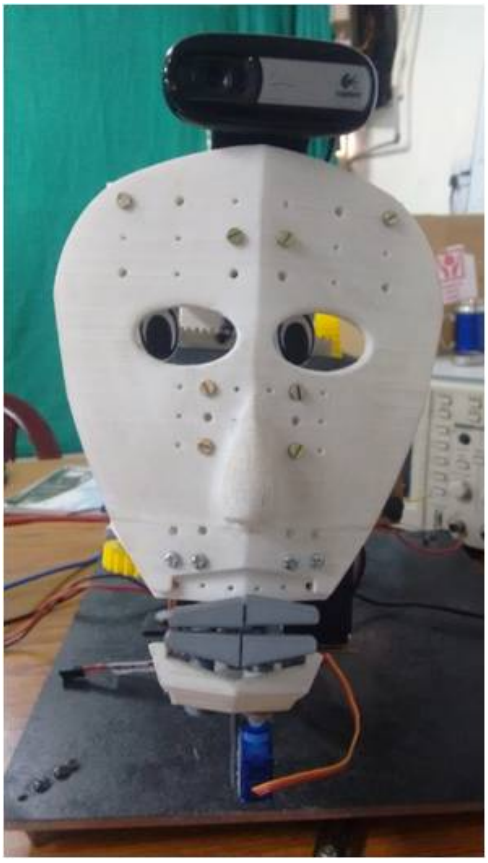

Figure 3 – Front View of the fabricated model

1.1 Neck

The neck is comprised of two DOFs for pan and tilt motions. The pan motion is in the transverse or horizontal plane which is powered by a high power DC servo motor. Since the pan motor has to drive the complete weight of the face acting parallel to the axis of rotation, the motor is required to deliver high torque. The tilt motion is in the sagittal plane of the robot and is driven by a PMDC motor operating as a closed loop system.Since this motion of the neck must tolerate the weight of the whole face as direct torque with a long lever arm, the otherwise used servo motors were discarded primarily because for our requirements, the servomotors would have been extremely costly.Thus, we resorted to using low cost PMDC motor providing it position feedback from a potentiometer based encoder. The final fabricated pictures of the robot with visible neck orientations are shown in Figure 3 and 4 respectively.

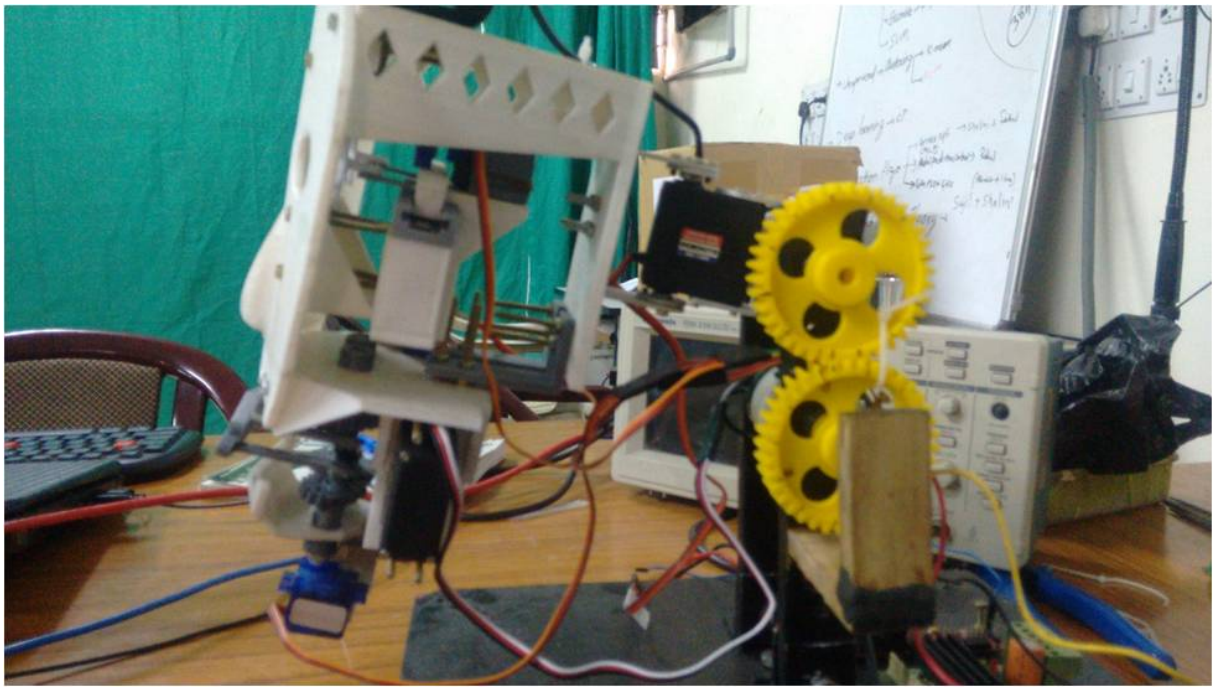

Figure 4 – Side View of the fabricated model with the visible neck mechanism

1.2 Jaw

The animatronic jaw is a 1-DOF arrangement shaped in the form of a human jaw. The jaw, being the only movable joint is driven by a medium torque servo motor because the weight of the same. The jaw also houses the mechanism for the movable lower lips. The motion is facilitated by a normal revolute joint where the end of the jaw is screwed to the star-head of the servo motor. The structure of the jaw has multiple faces along different planes resembling a pyramidal tapered cuboidal shape and does not have any housing for the human teeth. Figure 5 and 6 shows the close-up of the jaw – Front and side view.

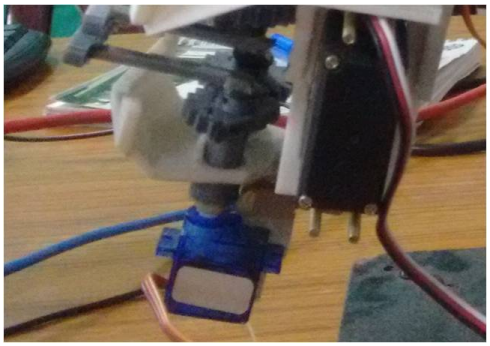

Figure 5 – Side View of the Jaw in the fabricated model

1.3 Eyes

The human eyeballs are able to move in frontal as well as transverse plane. However, for our model; we have only kept motion in the frontal plane to keep reduce the complexity of the design. The eyes of the robot are driven by a rack and pinion arrangement that drives the eyes right and left in co-ordination just as a normal human eye. The rack-pinion arrangement is driven by a low-torque servo motor with the pinion mounted on its top. The eye-balls are dummy and have no motion.

1.4 Face

The face of the robot is made out of ABS-plastic material and resembles a human face. It has got slots for the eyes and also an arrangement to house the mechanism for moving the upper lips. The face has a cover to attach it to the two motors driving the neck's pan and tilt. It also houses the eyes' motion mechanism with the support for the motor mounting and the rack and pinion arrangement. For giving computer vision, the face also has a camera mounted on the top. The complete design was firstly prepared on a virtual platform and then fabricated using 3D printer.

1.5 Lips

The lips of the robot are divided into two sections – the upper lips and the lower lips. The upper lips are attached with the face whereas the lower lips are mounted on the jaw. Each of the lips is divided into two sections that converge or diverge together from the central axis of their motion. It resembles the motion of the human lips in the horizontal direction. A servo motor driving two spur gear arrangement with each half of the lips connected to each of the gear facilitates coordinated motion to each of them. The upper lips are suspended from the face and protrude downwards whereas the lower lips are mounted on the inside of the jaw and can move along with the jaw, just as human lips do. Characteristics of all DOFs are summarized in Table 1 . As it can be seen, only two types of motors were used in this project. The different ratings of the motors are as per the requirements of the joint/motion that they power.

DOF	Motion Range	Motor Type	Reso.	Transmission	Motor Rating
Neck Pan	+90°to -90°	DC Servo	0.60°	Direct Coupling	8.4V, 16 Kgcm
Neck Tilt	+45°to -45°	DC PM	1.34°	Gear Train(1:1)	12V, 30RPM
Eyes	+90°to -90°	DC Servo	0.60°	Rack and Pinion	5.2V, 1.2 Kgcm
Lower Lips	+45°to -45°	DC Servo	0.60°	Gear Train(1:1)	5.2V, 1.2 Kgcm
Upper Lips	+45°to -45°	DC Servo	0.60°	Gear Train(1:1)	5.2V, 1.2 Kgcm
Jaw	0°to +60°	DC Servo	0.60°	Direct Coupling	7.4V, 8 Kgcm

Table 1 – Ratings and specifications of the various motors

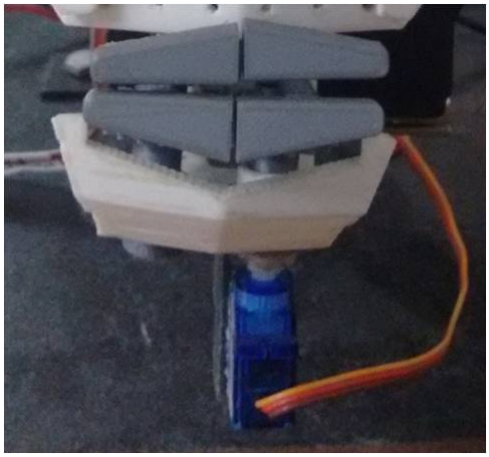

Figure 6 – Front View of the Jaw in the fabricated model

The facial expressions read by the perceiver have certain features that give the feel of different emotions from the face. Ekman and Friesen developed a commonly used facial measurement system called FACS. The system measures the face itself as opposed to trying to infer the underlying emotion given a particular facial configuration. This is a comprehensive system that distinguishes all possible visually distinguishable facial movements. Every such facial movement is the result of muscle action. Based on a deep understanding of how much muscle contraction changes visible appearance, it is possible to decompose any facial movement into anatomically minimal action units. FACS has defined 33 distinct action units (AUs) for the human face. Using the FACS system, we have compiled mappings of FACS action units to the expressions corresponding to anger, fear, happiness, surprise, disgust, and sadness based on the observations of different researchers. The Table 2 shown below shows the contribution of the various parts of the face in various expressions

	Happiness	Surprise	Anger	Disgust	Fear	Sadness
Eyebrows Frown			X	X	X	X
Raise Eyebrows		X			X	X
Raise Upper Eyelid		X	X		X	
Raise Lower Eyelid	X		X	X		
Up Turn Lip Corners	X					
Down Turn Lip Corners						X
Open Mouth	X	X			X	
Raise Upper Lip				X		

Table 2 – Facial Expression and their contributing face parts

The servo motors have been largely involved in the actuator systems of all the joints. Our robot utilizes a number of joints and links in its face to adjust its gaze direction and move its facial features. The actuator behind each of these joints is either a simple PMDC motor or a servo motor. Depending on the robot's perceptual data and emotional state, its brain issues appropriate motor commands for displaying facial expressions physically. This helps the robot to influence people emotionally and in a natural way. However, controlling robot's joints is more complicated than merely issuing simple positioning commands. If the actuators were linear and static, the environment was noise-free and disturbance did not exist, it would be ok. But a DC motor is a nonlinear dynamical actuator subjected to noise and disturbance. Therefore, a controller must be able to generate appropriate motor signals according to simple delivered motor commands. Usual controllers are based on a simplified model of the actual system. This is either due to our incomplete knowledge about the underlying mechanism and environment (e.g. noise) or accepting less accurate control to achieve a simpler design task. Regardless of the source of imprecision, it causes an error between the actual output and the desired output of the controller. However, measuring this error is usually less costly than obtaining an accurate model of the system or environment. Controllers that measure the actual output of the system and feed it back to the controller are called Closed-Loop or Feedback Controllers.

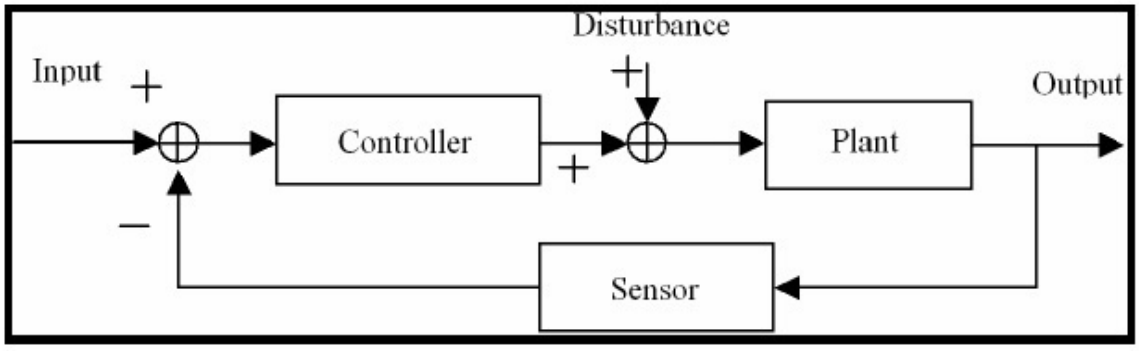

Figure 7 – Closed Loop Control System of a Servo Motor

However, for the PMDC motor used here - we have still used it as a servo motor by taking feedback from potentiometer based position encoders. Using software based PID control, we have managed to achieve satisfactory position control setup for the DC motor. In a closed-loop control system, the feedback signal is usually compared with the desired output to generate the appropriate control signal. A simple and common comparison criterion is the difference of these quantities, which is called error. In such controllers, they must then generate outputs that will decrease the absolute value of the error. A classic example in control theory for such an error-based controller is a PID controller. It has been successfully applied to motor control problem, both in theory and practice. A PID controller feeds to the plant a signal equal to a linear combination of the error, its integral and its derivative.

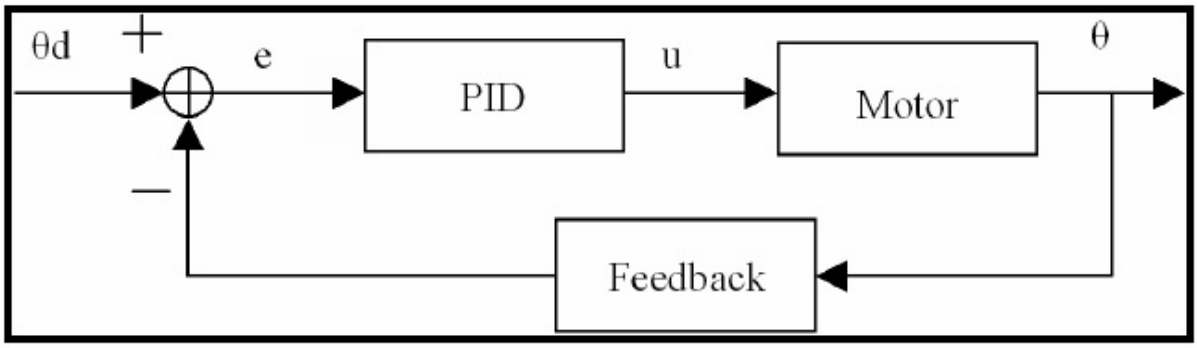

Figure 8 – Implementation of PID controller in a closed loop system

The hardware(software as well as other mechanical sections)components used in the robot include:

1. **DC Servo Motor** : A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists

of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. Servomotors are not a specific class of motor although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system as shown in the figure 9 below.

Figure 9 – DC Servo Motor

2. **Permanent Magnet DC Motor** : Permanent magnets (located in stator) provide magnetic field, instead of stator winding. The stator is usually made from steel in cylindrical form. Permanent magnets are usually made from rare earth materials or neodymium. These motors have comparatively easier control and operation than the other motors with variable field. We used a extended gear train for torque conversion to the PMDC motors which generally look as shown in the figure 10 below.

Figure 10 – Permanent Magnet DC Motor

3. **PCA 9685 – 16-channel 12-bit PWM Generator Module** : The PCA9685 is an I^2C-bus controlled 16-channel LED controller optimized for Red/Green/Blue/Amber

(RGBA) color backlighting applications. Each LED output has its own 12-bit resolution (4096 steps) fixed frequency individual PWM controller that operates at a programmable frequency from a typical of 24 Hz to 1526 Hz with a duty cycle that is adjustable from 0 % to 100 % to allow the LED to be set to a specific brightness value. All outputs are set to the same PWM frequency. A specimen is shown in Figure 11.

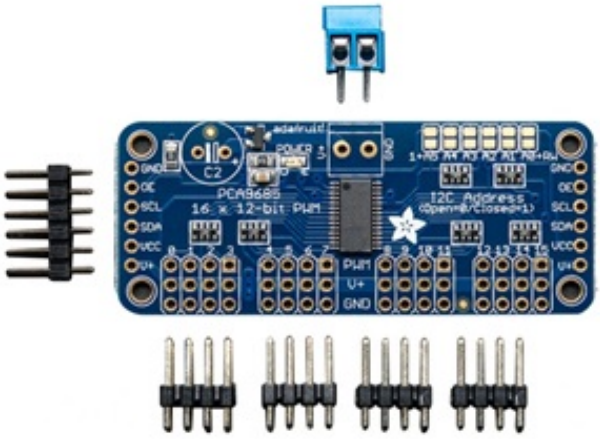

Figure 11 – PCA 9685 Module

4. **ADS 1115 – 4-channel 16-bit ADC module** : The ADS1115 device is a precision, low-power, 16-bit, I2C-compatible, analog-to-digital converter(ADC) offered in an ultra-small, leadless, X2QFN-10 package, and a VSSOP-10 package as shown in Figure 12 . The ADS1115 device incorporates a low-drift voltage reference and an oscillator. The ADS1114 and ADS1115 also incorporate a programmable gain amplifier (PGA) and a digital comparator. These features, along with a wide operating supply range, make the ADS111x well suited for power- and space-constrained, sensor measurement applications.

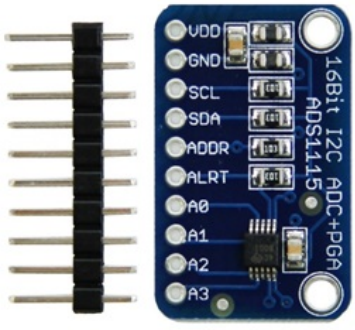

Figure 12 – ADS1115 ADC Module

The ADS1115 performs conversions at data rates up to 860 samples per second (SPS). The PGA offers input ranges from +- 256 mV to +- 6.144 V, allowing precise large and small-signal measurements. The ADS1115 features an input multiplexer (MUX) that allows two differential or four single-ended input measurements. Use the digital comparator in the ADS1114 and ADS1115 for under and over-voltage detection. The ADS1115 operates in either continuous-conversion mode or single-shot mode. The devices are automatically powered down after one conversion in single-shot mode; therefore, power consumption is significantly reduced during idle periods.

5. **L298D – Single Channel DC Motor Driver with PWM feature** : It is a dual H-bridge circuit used as a motor driver IC shown in figure 13, for high current applications where the current ratings are nearly 2 Amps or more.

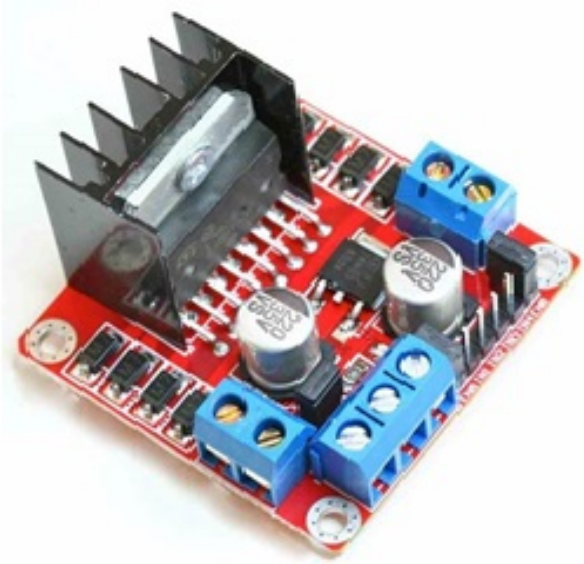

Figure 13 – Motor Driver Module

6. **11.1 V – 3 cell Lipo Battery** : A lithium polymer battery, or more correctly lithium-ion polymer battery (abbreviated variously as LiPo, LIP, Li-poly and others), is a rechargeable battery of lithium-ion technology using a polymer electrolyte instead of the more common liquid electrolyte. High conductivity semisolid (gel) polymers form the electrolyte for LiPo cells that are being used in tablet computers and many cellular telephone handsets with a specimen as shown in figure 14.

Figure 14 – A standard 5200mAh Lipo Battery

7. **22K Potentiometer for angular feedback**

8. **Raspberry Pi – The brain of the robot, LINUX based microprocessor** : The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside of its target market for uses such as robotics. Peripherals (including keyboards, mice and cases) are not included with the Raspberry Pi. Some accessories however have been included in several official and unofficial bundles.The Figure 15 shows a standard RPi 2 Model B.

Figure 15 – A Raspberry Pi 2 Model B

9. **Logitech Microphone integrated Webcam** :

   The robot actually communicates to the user using two interfaces as mentioned earlier – voice based interactive system and computer vision system. Thus, the webcam acts as the eyes of the robot that gives it a visual perception of the surroundings and the microphone is used to feed the voice commands. The processing of the computer vision feed and the speech is done on another dedicated machine and not the Raspberry Pi owing to the low processing power of the same. Thus, complete setup communicates

with the other software running on the other machines via socket programming. It used the standard TCP/IP based socket programming protocol where the Raspberry Pi acts as the server where it receives the tasks to be performed based on the inputs from the camera and the microphone. It also sends results of the tasks it performs and the other data that is needed by the other machines for processing the input and deciding onto next outputs. This Ethernet based communication system provides a maximum data transfer speed of 100 Gpbs.

Figure 16 – Microphone Integrated Webcam

Part II

Question Answering Module

2 Introduction

Enabling computers to understand given documents and answer questions about their content has recently attracted intensive interest. The recent availability of relatively large training datasets (Facebook babi task (1)) has made it more feasible to train and estimate rather complex models in an end-to-end fashion for these problems, in which a whole model is fit directly with given question-answer tuples and the resulting model has shown to be rather effective.

Our QA Module has primarily been divided in two parts, Visual based answering and Audio based answering systems.

Visual based system focusses on factoid answering. As mentioned earlier, our system has been designed to detect and recognize human faces from the input camera feed. Based on our learning algorithms and image processing techniques, the Actroid is able to answer basic quantitative questions. For example, the Actroid can easily find out the number of humans standing in front and can figure out their relative distances with itself based on the size of the face. It can then follow commands to turn its neck towards the person mentioned earlier. “Turn your head towards the person standing on your extreme left”. Distributed vector representation (2) helps in perfect execution of the command as well, as it manages slight variations in the sample command statement.

Audio based answering system is a four step process. The microphones attached to the system takes in human speech signal which then converted to text via Google’s API for speech to text conversion. The text question is then sent to QA architecture based on end to end memory networks which returns a text output. This text output is finally converted to speech via Google’s API for text to speech conversion and the answer is heard through the connected speakers.

We use a neural network with a recurrent attention model over a possibly large external memory. The architecture is a form of Memory Network but unlike the model in that work, it is trained end-to-end, and hence requires significantly less supervision during training, making it more generally applicable in realistic settings. It can also be seen as an extension of RNNsearch to the case where multiple computational steps (hops) are performed per output symbol. The flexibility of the model allows us to apply it to tasks as diverse as (synthetic) question answering and to language modeling. The questions are divided in five major categories, covering multiple genres of biographical information. Our system currently supports all 20 types of babi question answering tasks.

The main idea of babi tasks (3) is to provide a set of tasks, in a similar way to how software testing is built in computer science. Ideally each task is a “leaf” test

case, as independent from others as possible, and tests in the simplest way possible one aspect of intended behavior. Subsequent ("non-leaf") tests can build on these by testing combinations as well. Each task provides a set of training and test data, with the intention that a successful model performs well on test data. The supervision in the training set is given by the true answers to questions, and the set of relevant statements for answering a given question, which may or may not be used by the learner.

2.1 Babi task set

Here are the 20 genres defined in the babi tasks. (4)

1. **Single Supporting Fact** : Task 1 consists of questions where a previously given single supporting fact, potentially amongst a set of other irrelevant facts, provides the answer. We first test one of the simplest cases of this, by asking for the location of a person, e.g. "Mary travelled to the office. Where is Mary?"

2. **Two or Three Supporting Facts** : A harder task is to answer questions where two supporting statements have to be chained to answer the question, as in task 2, where to answer the question "Where is the football?" one has to combine information from two sentences "John is in the playground" and "John picked up the football".

3. **Two or Three Argument Relations** : To answer questions the ability to differentiate and recognize subjects and objects is crucial. In task 4 we consider the extreme case where sentences feature reordered words, i.e. a bag-of-words will not work. For example, the questions "What is north of the bedroom?" and "What is the bedroom north of?" have exactly the same words, but a different order, with different answers. A step further, sometimes one needs to differentiate three separate arguments. Task 5 involves statements like "Jeff was given the milk by Bill" and then queries who is the giver, receiver or which object is involved.

4. **Yes/No Questions** : Task 6 tests, on some of the simplest questions possible (specifically, ones with a single supporting fact), the ability of a model to answer true/false type questions like "Is John in the playground?"

5. **Counting and Lists/Sets** : Task 7 tests the ability of the QA system to perform simple counting operations, by asking about the number of objects with a certain property, e.g. "How many objects is Daniel holding?" Similarly, task 8 tests the ability to produce a set of single word answers in the form of a list, e.g. "What is Daniel holding?" These tasks can be seen as QA tasks related to basic database search operations.

6. **Simple Negation and Indefinite Knowledge** : Tasks 9 and 10 test slightly more complex natural language constructs. Task 9 tests one of the simplest forms of negation that of supporting facts that imply a statement is false e.g. "Fred is no longer in the office" rather than "Fred travelled to the office". (In this case, task 6 (yes/no questions) is a prerequisite to the task.) Task 10 tests if we can model statements that describe possibilities rather than certainties, e.g. "John is either in the classroom or the playground." where in that case the answer is "maybe" to the question "Is John in the classroom?"

7. **Basic Coreference, Conjunctions and Compound Coreference** : Task 11 tests the simplest type of Coreference, that of detecting the nearest referent, e.g. "Daniel was in the kitchen. Then he went to the studio." Real-world data typically addresses this as a labeling problem and studies more sophisticated phenomena, whereas we evaluate it as in all our other tasks as a question answering problem. Task 12 (conjunctions) tests referring to multiple subjects in a single statement, e.g. "Mary and Jeff went to the kitchen.". Task 13 tests coreference in the case where the pronoun can refer to multiple actors, e.g. "Daniel and Sandra journeyed to the office. Then they went to the garden".

8. **Time Reasoning** : While our tasks so far have included time implicitly in the order of the statements, task 14 tests understanding the use of time expressions within the statements, e.g. "In the afternoon Julie went to the park. Yesterday Julie was at school.", followed by questions about the order of events such as "Where was Julie before the park?". Real-world datasets address the task of evaluating time expressions typically as a labeling, rather than a QA task.

9. **Basic Deduction and Induction** : Task 15 tests basic deduction via inheritance of properties, e.g. "Sheep are afraid of wolves. Gertrude is a sheep. What is Gertrude afraid of?". Task 16 similarly tests basic induction via inheritance of properties. A full analysis of induction and deduction is clearly beyond the scope of this work, and future tasks should analyse further, deeper aspects.

10. **Positional and Size Reasoning** : Task 17 tests spatial reasoning, one of many components of the classical SHRDLU system by asking questions about the relative positions of colored blocks. Task 18 requires reasoning about the relative size of objects and is inspired by the commonsense reasoning examples in the Winograd schema challenge.

11. **Path Finding** : The goal of task 19 is to find the path between locations: given the description of various locations, it asks: how do you get from one to another? This is related to the work of Chen & Mooney (2011) and effectively involves a search problem.

12. **Agent's Motivations** : Finally, task 20 questions, in the simplest way possible, why an agent performs an action. It addresses the case of actors being in a given state (hungry, thirsty, tired, . . .) and the actions they then take, e.g. it should learn that hungry people might go to the kitchen, and so on.

As it will be described in the later stages of our report, end to end memory system works on limited vocabulary, ie. The words present in training set dictionary can only be used to frame test sentences.

3 End to End Memory Networks

The architecture is a form of Memory Network but unlike the model in that work, it is trained end-to-end, and hence requires significantly less supervision during training, making it more generally applicable in realistic settings. It can also be seen as an extension of RNNsearch to the case where multiple computational steps (hops) are performed per output symbol. The flexibility of the model allows us to apply it to tasks as diverse as (synthetic) question answering and to language modeling.

Two grand challenges in artificial intelligence research have been to build models that can make multiple computational steps in the service of answering a question or completing a task, and models that can describe long term dependencies in sequential data. Recently there has been a resurgence in models of computation using explicit storage and a notion of attention; manipulating such a storage offers an approach to both of these challenges. In multiple papers, the storage is endowed with a continuous representation; reads from and writes to the storage, as well as other processing steps, are modeled by the actions of neural networks.

In this work (4), we present a novel recurrent neural network (RNN) (5) architecture where the recurrence reads from a possibly large external memory multiple times before outputting a symbol. This model can be considered a continuous form of the Memory Network implemented in the mentioned reference paper. The model in that work was not easy to train via backpropagation, and required supervision at each layer of the network. The continuity of the model we present here means that it can be trained end-to-end from input-output pairs, and so is applicable to more tasks, i.e. tasks where such supervision is not available, such as in language modeling or realistically supervised question answering tasks. This model can also be seen as a version of RNNsearch with multiple computational steps (which we term "hops") per output symbol. It has been shown experimentally that the multiple hops over the long-term memory are crucial to good performance of our model on these tasks, and that training the memory representation can be integrated in a scalable manner into our end-to-end neural network model.

The model takes a discrete set of inputs $x_1, ..., x_n$ that are to be stored in the memory, a query q, and outputs an answer a. Each of the x_i , q, and a contains symbols coming from a dictionary with V words. The model writes all x to the memory up to a fixed buffer size, and then finds a continuous representation for the x and q. The continuous representation is then processed via multiple hops to output a. This allows backpropagation (6) of the error signal through multiple memory accesses back to the input during training.

3.1 Single Layer

We start by describing our model in the single layer case, which implements a single memory hop operation. We then show it can be stacked to give multiple hops in memory.

1. **Input Memory Representation** : Suppose we are given an input set $x_1, ..., x_i$ to be stored in memory. The entire set of x_i are converted into memory vectors m_i of dimension d computed by embedding each x_i in a continuous space, in the simplest case, using an embedding matrix A (of size $d * V$). The query q is also embedded (again, in the simplest case via another embedding matrix B with the same dimensions as A) to obtain an internal state u. In the embedding space, we compute the match between u and each memory mi by taking the inner product followed by a softmax:

$$p_i = softmax(u^T m_i) \tag{3.1}$$

where $softmax(z_i) = e^{z_i} / \sum_j e^{z_j}$. Defined in this way p is a probability vector over the inputs.

2. **Output Memory Representation** : Each x_i has a corresponding output vector c_i (given in the simplest case by another embedding matrix C). The response vector from the memory o is then a sum over the transformed inputs c_i , weighted by the probability vector from the input:

$$o = \sum_i p_i c_i \tag{3.2}$$

Because the function from input to output is smooth, we can easily compute gradients and backpropagate through it.

3. **Generating the final prediction** : In the single layer case, the sum of the output vector o and the input embedding u is then passed through a final weight matrix W (of size $V * d$) and a softmax to produce the predicted label :

$$a^{'} = softmax(W(o + u)) \tag{3.3}$$

The overall model is shown in Figure 3.1(a) below. During training, all three embedding matrices A, B and C, as well as W are jointly learned by minimizing a standard cross-entropy loss between a' and the true label a. Training is performed using stochastic gradient descent.

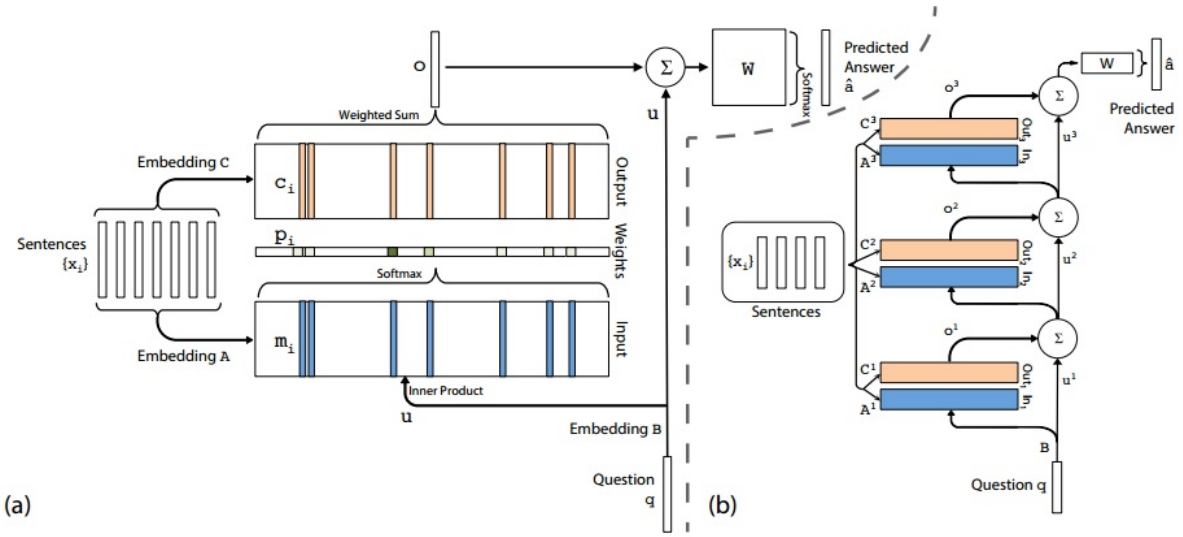

Figure 17 – Figure (a) and (b) describes the single and three layer version of the model

3.2 Multiple Layers

We now extend our model to handle K hop operations. The memory layers are stacked in the following way:

1. The input to layers above the first is the sum of the output o^k and the input u^k from layer k (different ways to combine o^k and u^k are proposed later):

$$u^{k+1} = u^k + o^k \tag{3.4}$$

2. Each layer has its own embedding matrices A^k , C^k , used to embed the inputs x_i. However, as discussed below, they are constrained to ease training and reduce the number of parameters.

3. At the top of the network, the input to W also combines the input and the output of the top memory layer:

$$a^{'} = softmax(Wu^{K+1}) = softmax(W(o^K + u^K)) \tag{3.5}$$

We explore two types of weight tying within the model:

1. **Adjacent** : The output embedding for one layer is the input embedding for the one above, i.e. $A^{k+1} = C^k$. We also constrain

   a. The answer prediction matrix to be the same as the final output embedding, i.e $W^T = C^K$, and

b. The question embedding to match the input embedding of the first layer, i.e. $B = A^1$.

2. **Layer-wise RNN like** : The input and output embeddings are the same across different layers, i.e. $A^1 = A^2 = ... = A^K$ and $C^1 = C^2 = ... = C^K$. We have found it useful to add a linear mapping H to the update of u between hops; that is, $u^{k+1} = Hu^k + o^k$. This mapping is learnt along with the rest of the parameters and used throughout our experiments for layer-wise weight tying.

A three-layer version of our memory model is shown in Figure (b). Overall, it is similar to the Memory Network model in, except that the hard max operations within each layer have been replaced with a continuous weighting from the softmax.

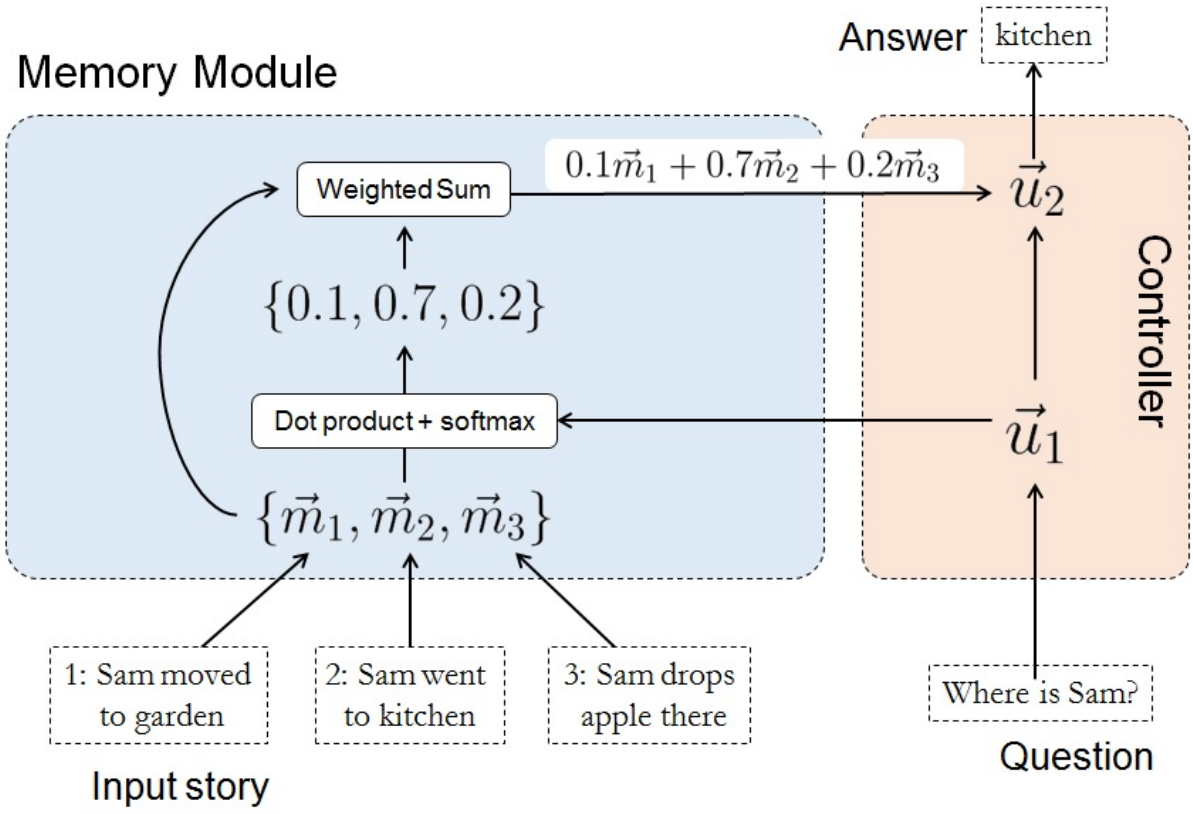

Figure 18 – MemN2N in working

Note that if we use the layer-wise weight tying scheme, our model can be cast as a traditional RNN where we divide the outputs of the RNN into internal and external outputs. Emitting an internal output corresponds to considering a memory, and emitting an external output corresponds to predicting a label. From the RNN point of view, u and the equation mentioned above is a hidden state, and the model generates an internal output p using A. The model then ingests p using C, updates the hidden state, and so on.

Here, unlike a standard RNN, we explicitly condition on the outputs stored in memory during the K hops, and we keep these outputs soft, rather than sampling them. Thus our model makes several computational steps before producing an output meant to be seen by the "outside world".

The figure below describes the approach followed for the audio based answering system.

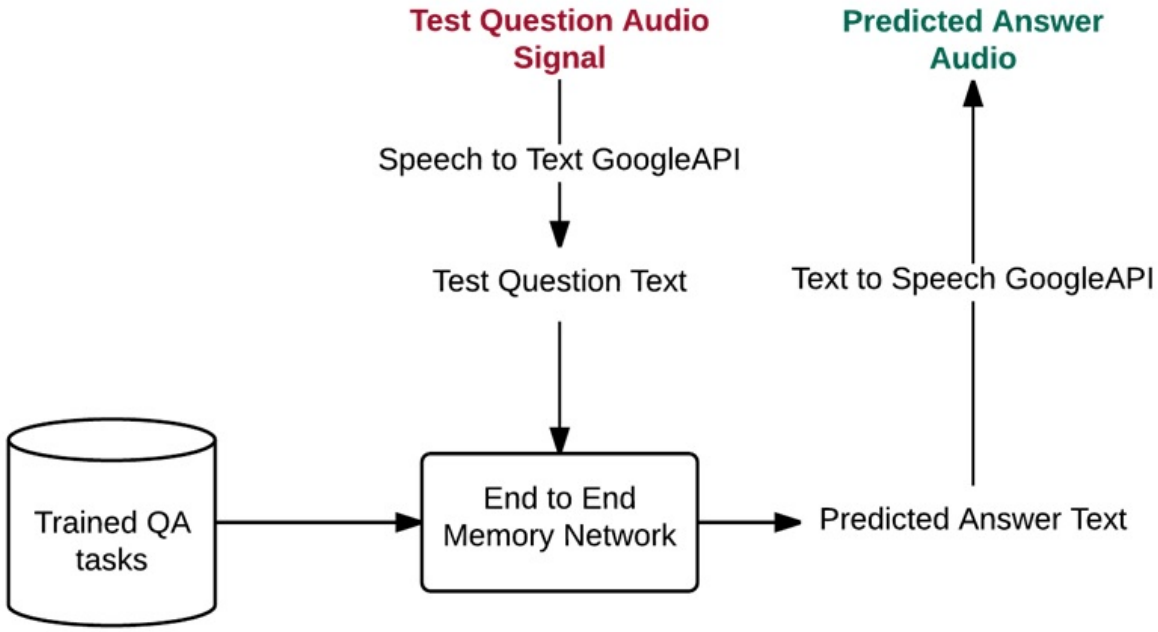

Figure 19 – Audio Based Answering System Approach

3.3 Related Work

A number of recent efforts have explored ways to capture long-term structure within sequences using RNNs or LSTM-based models. The memory in these models is the state of the network, which is latent and inherently unstable over long timescales. The LSTM-based models address this through local memory cells which lock in the network state from the past. In practice, the performance gains over carefully trained RNNs are modest (see Mikolov et al.). Our model differs from these in that it uses a global memory, with shared read and write functions. However, with layer-wise weight tying our model can be viewed as a form of RNN which only produces an output after a fixed number of time steps (corresponding to the number of hops), with the intermediary steps involving memory input/output operations that update the internal state.

Some of the very early work on neural networks by Steinbuch and Piske and Taylor considered a memory that performed nearest-neighbor operations on stored input vectors

and then fit parametric models to the retrieved sets. This has similarities to a single layer version of our model.

Subsequent work in the 1990's explored other types of memory. For example, Das et al. and Mozer et al. introduced an explicit stack with push and pop operations which has been revisited recently by in the context of an RNN model.

Closely related to our model is the Neural Turing Machine of Graves et al., which also uses a continuous memory representation. The NTM memory uses both content and address-based access, unlike ours which only explicitly allows the former, although the temporal features that we will introduce in Section 4.1 allow a kind of address-based access. However, in part because we always write each memory sequentially, our model is somewhat simpler, not requiring operations like sharpening. Furthermore, we apply our memory model to textual reasoning tasks, which qualitatively differ from the more abstract operations of sorting and recall tackled by the NTM.

Our model is also related to Bahdanau et al. In that work, a bidirectional RNN based encoder and gated RNN based decoder were used for machine translation. The decoder uses an attention model that finds which hidden states from the encoding are most useful for outputting the next translated word; the attention model uses a small neural network that takes as input a concatenation of the current hidden state of the decoder and each of the encoders hidden states. A similar attention model is also used in Xu et al. for generating image captions. Our "memory" is analogous to their attention mechanism, although it is only over a single sentence rather than many, as in our case. Furthermore, our model makes several hops on the memory before making an output; we will see below that this is important for good performance. There are also differences in the architecture of the small network used to score the memories compared to our scoring approach; we use a simple linear layer, whereas they use a more sophisticated gated architecture.

We will apply our model to language modeling, an extensively studied task. Goodman showed simple but effective approaches which combine n-grams with a cache. Bengio et al. ignited interest in using neural network based models for the task, with RNNs and LSTMs showing clear performance gains over traditional methods. Indeed, the current state-of-the-art is held by variants of these models, for example very large LSTMs with Dropout or RNNs with diagonal constraints on the weight matrix. With appropriate weight tying, our model can be regarded as a modified form of RNN, where the recurrence is indexed by memory lookups to the word sequence rather than indexed by the sequence itself.

4 Synthetic Question and Answering Experiments

We perform experiments on the synthetic QA tasks defined in the babi task set (using version 1.1 of the dataset). A given QA task consists of a set of statements, followed by a question whose answer is typically a single word (in a few tasks, answers are a set of words). The answer is available to the model at training time, but must be predicted at test time. There are a total of 20 different types of tasks that probe different forms of reasoning and deduction. Here are samples of three of the tasks:

1. Sam walks into the kitchen. Sam picks up an apple. Sam walks into the bedroom. Sam drops the apple.

   Where is the apple? Bedroom

2. Brian is a lion. Julius is a lion. Julius is white. Bernhard is green.

   What colour is Brian? White

3. Mary journeyed to the den. Mary went back to the kitchen. John journeyed to the bedroom. Mary discarded the milk.

   Where was the milk before the den? Hallway

Note that for each question, only some subset of the statements contain information needed for the answer, and the others are essentially irrelevant distractors (e.g. the first sentence in the first example). In the Memory Networks of Weston et al., this supporting subset was explicitly indicated to the model during training and the key difference between that work and this one is that this information is no longer provided. Hence, the model must deduce for itself at training and test time which sentences are relevant and which are not.

Formally, for one of the 20 QA tasks, we are given example problems, each having a set of I sentences xi where I <= 320; a question sentence q and answer a. Let the j^{th} word of sentence i be x_{ij} , represented by a one-hot vector of length V (where the vocabulary is of size V = 177, reflecting the simplistic nature of the QA language). The same representation is used for the question q and answer a. Two versions of the data are used, one that has 1000 training problems per task and a second larger one with 10,000 per task.

Model Details : Unless otherwise stated, all experiments used a $K = 3$ hops model with the adjacent weight sharing scheme. For all tasks that output lists (i.e. the

answers are multiple words), we take each possible combination of possible outputs and record them as a separate answer vocabulary word.

1. **Sentence Representation** : In our experiments we explore two different representations for the sentences. The first is the bag-of-words (BoW) representation that takes the sentence $x_i = x_{i1}, x_{i2}, ..., x_{in}$, embeds each word and sums the resulting vectors: example $m_i = \sum_j Ax_{ij}$ and $c_i = \sum_j Cx_{ij}$. The input vector u representing the question is also embedded as a bag of words: $u = \sum_j Bqj$. This has the drawback that it cannot capture the order of the words in the sentence, which is important for some tasks.

2. **Temporal Encoding** : Many of the QA tasks require some notion of temporal context, i.e. in the first example mentioned above, the model needs to understand that Sam is in the bedroom after he is in the kitchen. To enable our model to address them, we modify the memory vector so that $m_i = \sum_j Ax_{ij} + TA(i)$ where $TA(i)$ is the i^{th} row of a special matrix TA that encodes temporal information.

The output embedding is augmented in the same way with a matrix TC. Both TA and TC are learned during training. They are also subject to the same sharing constraints as A and C. Note that sentences are indexed in reverse order, reflecting their relative distance from the question so that x_1 is the last sentence of the story.

4.1 Training Details

10 % of the bAbI training set was held-out to form a validation set, which was used to select the optimal model architecture and hyperparameters. Our models were trained using a learning rate of $\eta = 0.01$, with anneals every 25 epochs by $\eta/2$ until 100 epochs were reached. No momentum or weight decay was used. The weights were initialized randomly from a Gaussian distribution with zero mean and $\sigma = 0.1$. When trained on all tasks simultaneously with 1k training samples (10k training samples), 60 epochs (20 epochs) were used with learning rate anneals of $\eta/2$ every 15 epochs (5 epochs). All training uses a batch size of 32 (but cost is not averaged over a batch), and gradients with an ‘2 norm larger than 40 are divided by a scalar to have norm 40. In some of our experiments, we explored commencing training with the softmax in each memory layer removed, making the model entirely linear except for the final softmax for answer prediction. When the validation loss stopped decreasing, the softmax layers were re-inserted and training recommenced. We refer to this as linear start (LS) training. In LS training, the initial learning rate is set to $\eta = 0.005$. The capacity of memory is restricted to the most recent 50 sentences. Since the number of sentences and the number of words per sentence varied between problems, a

null symbol was used to pad them all to a fixed size. The embedding of the null symbol was constrained to be zero.

On some tasks, we observed a large variance in the performance of our model (i.e. sometimes failing badly, other times not, depending on the initialization). To remedy this, we repeated each training 10 times with different random initializations, and picked the one with the lowest training error.

The described model has achieved maximum possible accuracies in babi tasks 1 to 3 in less than 8 iterations over the 10k training examples.

Part III

Computer Vision System

5 Face detection and Tracking

5.1 Introduction

The visual perceptive intelligence of the Actroid system is majorly dealt by the Computer Vision (CV) technology. Not only should the Actroid see its surroundings, but it should also extract maximal information from and thereby interact with the surroundings just like any other human. The surroundings may comprise of living and non-living things. CV should enable the system to recognize different worldly objects, like pencil, notebook, bottle, etc., and memorize human faces exhibiting memory better than any other human. The proactive system would discern any relevant activity in the vicinity and form experience-based knowledge which could be updated frequently. With artificially integrated intelligence, the possibilities regarding the applications of the system are endless. In the project, the objective of the computer vision system is to achieve efficient face detection and tracking, face recognition and object recognition.

Face detection is the first step in face recognition process which involves locating the human faces in a digital image or video frame. Face detection is one of the challenging task in computer vision field because of variability in human faces. A robust face detector should be able to find faces in frame regardless of their number, orientation, pose and expression. Lots of work is going on in this area to improve the accuracy and speed of detection. Just like we humans detect and recognize any object by looking at their features and characteristics, a computer also needs features for detecting object. This feature based detection is most common in face detection. These facial features are called descriptors. Two most common feature based face detection algorithm that exist are Viola-Jones algorithm and Histogram of Oriented Gradient (HOG).

5.2 Viola Jones Algorithm

One of the foundational work in the area of face detection was done by Viola and Jones in 2001 which involves detection of human faces in real time based on Haar like features. The algorithm is popularly known as Viola-Jones object tracking algorithm and is the first to realize robust object detection near to real-time. Here, the algorithm is used for frontal Face Detection purpose. For Face Detection in an input image, the Viola-Jones algorithm considers a 24x24 pixel window which is placed one-by-one all over the image. At every location, the window performs some processing to extract relevant information in the form of features on the basis of which the presence of anything that resembles a face is detected.The Viola-Jones algorithm was a breakthrough in object detection as it

brought together three seminal concepts namely - Integral Image, Adaboost training and Cascaded classifier. Below subsections describe the steps of Viola-Jones algorithm

5.2.1 Haar features selection

Features, in object detection, also known as kernels, are images with size less than the input image and which could be convolved over the input image to extract meaningful information. Figure 20 shows a 3x3 edge detection convolutional kernel where each white/black block represents one pixel. The convolution over the input image leads to the subtraction of the black part of the kernel from the white part. It can be inferred from Figure 20 that the shape of the kernel should resemble the feature that it is detecting in the image.

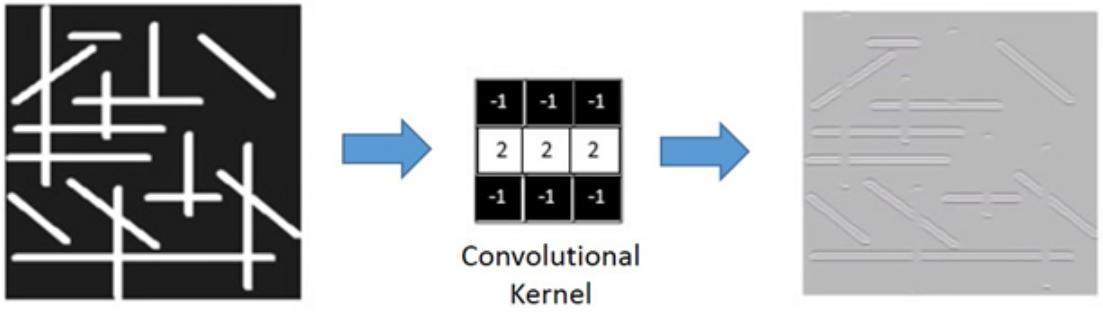

Figure 20 – Edge detection using convolutional kernel

Haar features are similar to these convolutional kernels which are used to detect the presence of that feature in the input image. For example, the presence of eyes above cheek can be detected by a rectangular haar feature having upper part as black and lower as white and the eyebrows can be detected using a haar feature having white part sandwiched by black parts as shown in Figure 21.

Figure 21 – Different types of Haar features

5.2.2 Integral Image

The calculation of the convolution of each Haar feature with the 24x24 window can be made quicker using the integral image representation of the original image. The

integral image at any pixel is the sum of pixels above and to the left of that pixel.

5.2.3 Adaboost Training

In a 24x24 pixel window, 160,000+ haar features can be formed. Calculating each feature in every window would prevent the algorithm to run in real time. Thus, Adaboost training is used to find out the most relevant features that would be sufficient for Face Detection. The relevant features are individually termed as weak classifiers and their weighted sum is termed as Strong classifier.

5.2.4 Cascading classifiers

The computational complexity of the face detection process can further be alleviated by cascading the aforementioned classifiers in such a fashion that non-faces can be discarded in fewer steps. A cascade classifier is composed of stages each composed of a set of features as shown in Figure 22. Stage 1 may contain the most relevant features whose absence should be a direct indication of a non-face. If the Stage 1 indicates the presence of the features, then the sample is forwarded to Stage 2 and the process continues till the last stage. Only those samples pass which actually contain a face. Therefore, for input images containing only two faces, all the stages will be processed for only those two windows and the rest would be discarded in earlier stages.

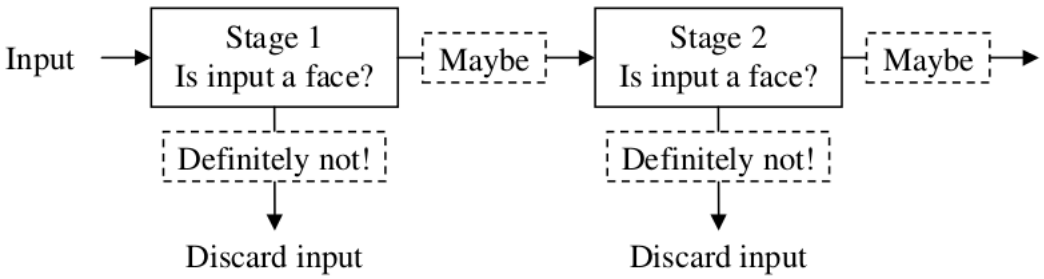

Figure 22 – Cascaded classifier

5.3 Histogram of Oriented Gradient (HOG)

HOG is a description method in which every point (pixel) or block of image is replaced by the orientation of gradients. These gradients (x and y derivatives) are nothing but directional change of intensity in an image. Replacing pixels with gradients make the detection process independent of illumination. Moreover gradients are useful as magnitude

of gradients is large around edges and corners where intensity changes abruptly. These edges and corners gives us lot more information about object than flat regions.

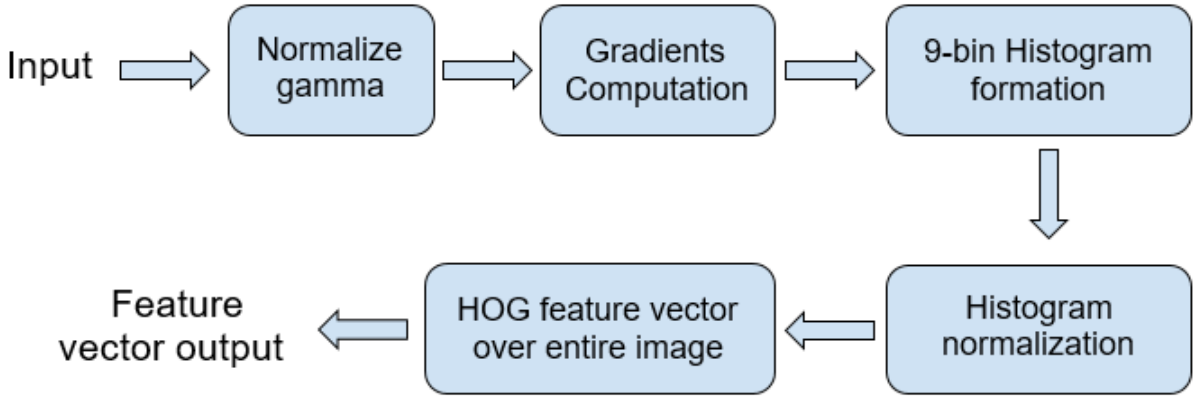

Figure 23 – Process flow of HOG feature extraction

5.3.1 Gamma Normalization

Gamma normalization is an image enhancement technique which improves the contrast of image to make it clearer. But this step is optional and can be omitted without losing performance.

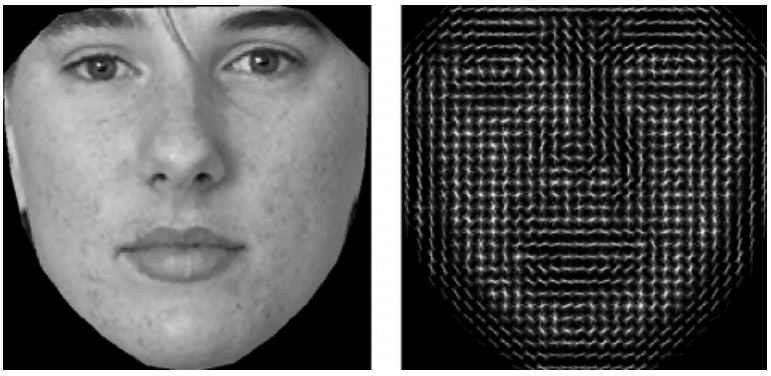

Figure 24 – Visualization of Gradients

5.3.2 Gradients Computation

Gradients are computed in x and y direction. This is easily achieved by filtering the image with following kernels $[-1, 0, 1]$ and $[-1, 0, 1]^T$.

5.3.3 Histogram of gradients computation

In this step, image is divided into 8x8 cells (as shown in Figure 25) and histogram of gradients is calculated for each 8x8 cells. The histogram is a vector of 9 bins (numbers) corresponding to angles $0^0, 20^0, 40^0, \ldots, 160^0$

Figure 25 – Dividing image in smaller cells

To further explain, let us look at one 8x8 cell in the Figure 26 and see how the gradients look. From Figure 26, image in center shows the gradients of 8x8 cell. Arrows point in the direction of gradient and the length of arrows shows its magnitude. Extreme right image shows the matrix of numbers representing gradients.

Next step is to create a histogram of gradients in these 8x8 cells. The histogram contains 9 bins corresponding to angles $0^0, 20^0, 40^0, \ldots, 160^0$

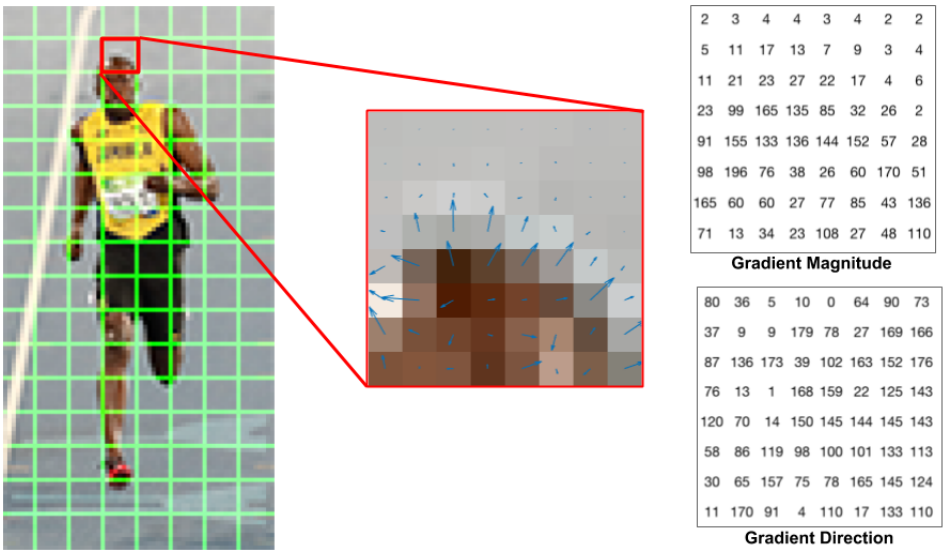

Figure 26 – Visualization of gradients magnitude and direction in a single cell

Figure 27 illustrates the process of making histogram. We are looking at magnitude and direction of the gradient of the same 8x8 patch as shown in the figure 26

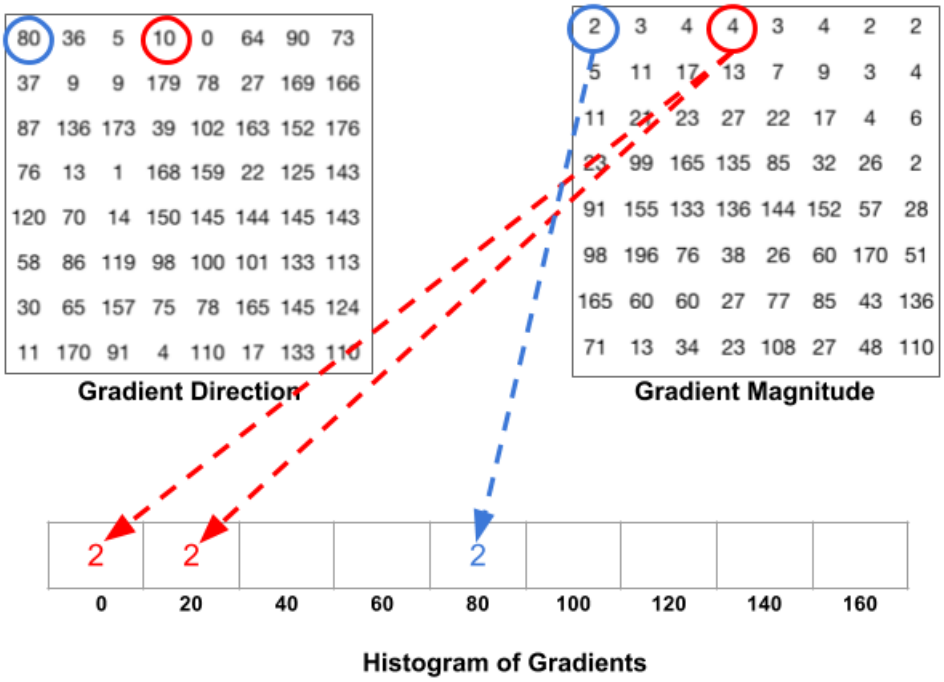

Figure 27 – Assigning votes to each bin

A bin is selected based on the direction, and the vote (the value that goes into the bin) is selected based on the magnitude. Let's first focus on the pixel encircled in blue. It has an angle (direction) of 80 degrees and magnitude of 2. So it adds 2 to the 5^{th} bin. The gradient at the pixel encircled using red has an angle of 10^0 and magnitude of 4. Since 10^0 is half way between 0 and 20, the vote by the pixel splits evenly into the two bins. The contributions of all the pixels in the 8x8 cells are added up to create the 9-bin histogram. For the patch above, it looks like figure 28

5.3.4 16x16 Block normalization

So far, we have the histogram based on the calculated gradients of image. But these gradients of image are sensitive to lighting conditions. To make it independent of lighting variations, normalization of histogram is done.To make the process somewhat faster, normalization is done over a bigger sized block of 16x16. A 16x16 block has 4 histograms which can be concatenated to form a 36 x 1 element vector. The window is then moved by 8 pixels and a normalized 36x1 vector is calculated over the window.

5.3.5 Final HOG feature vector

Final HOG feature vector is obtained by concatenating the all 36x1 vector of entire image. Original image was of 64x128 pixel. Each 16x16 blocks cover 7 horizontal positions and 15 vertical positions making a total of 7x15 = 105 blocks Each 16x16 block is represented by 36x1 normalized vector as explained in step 4. So concatenating all 105

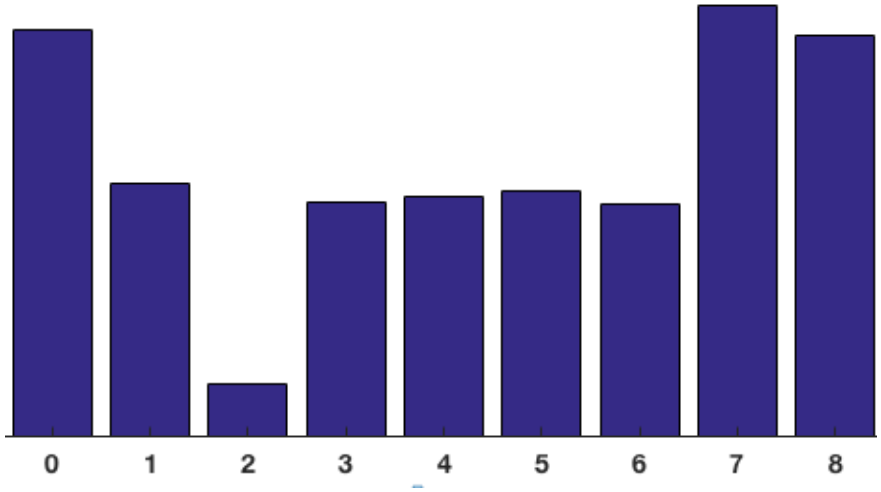

Figure 28 – Final 9-bin histogram of a single 8x8 cell

16x16 blocks, we obtain 105x36 = 3780 dimensional vector. So the final HOG feature vector obtained for an image of 64x128 pixel is of size 3780x1.

6 Face Recognition

6.1 Introduction

Face detection acts as primary step for face recognition. Once the window containing the face is detected various algorithms can be used to train the system for recognizing faces. Commonly, in order to train the system to recognize an individual, the process involves training on hundreds of samples images of that individual, after which it can be tested to accurately recognize the individual whenever he or she comes in front of the system.

Face recognition has been an active research topic since the 1970's. Given an input image with multiple faces, face recognition systems typically first run face detection to isolate the faces. Each face is preprocessed and then a low-dimensional representation (or embedding) is obtained. A low-dimensional representation is important for efficient classification. Challenges in face recognition arise because the face is not a rigid object and images can be taken from many different viewpoints of the face. Face representations need to be resilient to intra-personal image variations such as age, expressions, and styling while distinguishing between inter-personal image variations between different people.

Today's top-performing face recognition techniques are based on convolutional neural networks. Facebook's DeepFace (7), Microsoft's ResNet model (8) and Google's FaceNet (9) system yield the highest accuracy.

Figure 29 shows the logic flow for face recognition with neural networks. There are many face detection method to choose from, and we will be using our previous face detection method based on HOG face descriptors. Once face is detected, various preprocessing techniques can be used to normalize and make fixed-size faces to feed into neural network. Neural network is used as feature extractor to produce low dimensional representations that characterizes a person's face. Those low dimensional features are easy to train for either classification or clustering task.

In literature, the face recognition algorithms involving deep neural networks have a classification layer followed by a bottleneck layer which is supposed to generalize the recognition beyond the set of identities used in training (10, 7). This intermediate bottleneck layer results in indirectness and inefficiency and a large representation size per face (1000s of dimensions).

We will be implementing Google's FaceNet system for our face recognition task which is based on the FaceNet publications by Google researchers in 2015 (9). This paper introduces a novel CNN based architecture that learns to directly map a face image

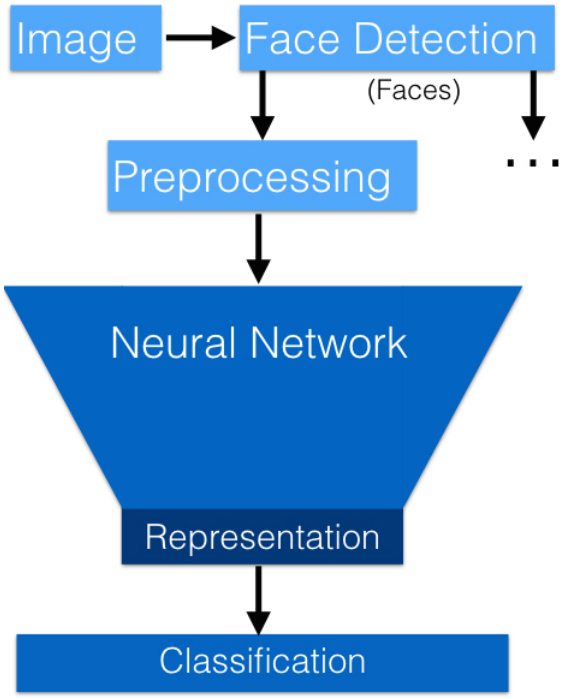

Figure 29 – Face Recognition with Neural Networks

in compact Euclidean space. The distances between representation vectors are a direct measure of their similarity. The representation significantly reduces the image complexity to 128-bytes per face. FaceNet is trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. Instead of the Inception model (11) presented in the FaceNet system, the ResNet deep convolution architecture (8) is used.

FaceNet's 128-D embeddings are trained against the output data using a triplet-based loss function based on LMNN (12). The triplets consist of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the positive pair from the negative by a distance margin. For the generation of triplet based training set, a novel online negative exemplar mining strategy is used which ensures consistently increasing difficulty of triplets as the network trains.

The approach exhibits incredible robustness against illumination and pose variations (Figure 30). The approach is a purely data driven method which learns its representation directly from the pixels of the face. Rather than using engineered features, we use a large dataset of labelled faces to attain the appropriate invariances to pose, illumination, and other variational conditions.

Subsequent sections cover the definition of the triplet loss function, the novel online negative exemplar mining strategy and the FaceNet deep network architecture.

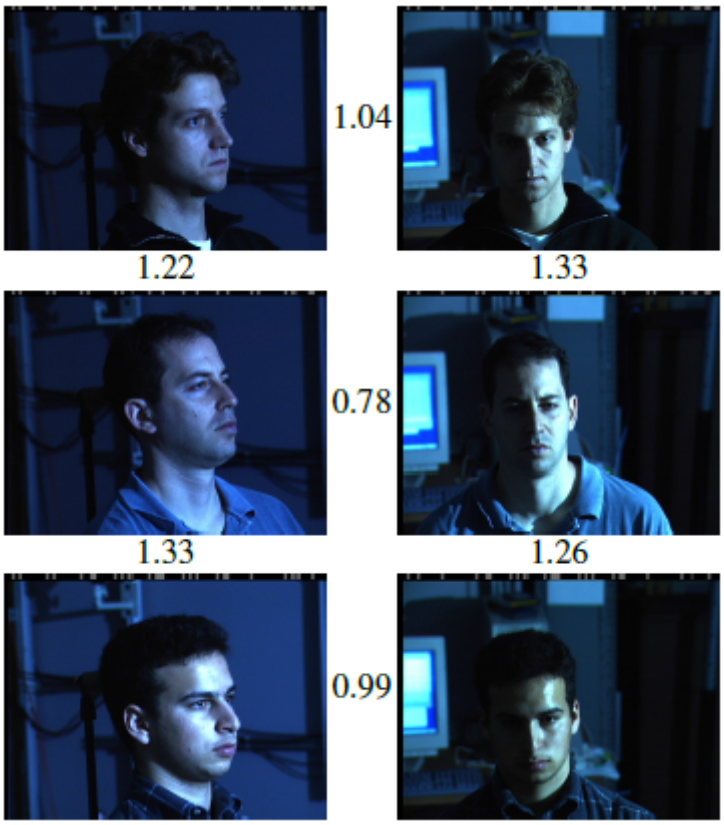

Figure 30 – Output distances of FaceNet between pairs of faces of the same and a different person in different pose and illumination combinations. A distance of 0:0 means the faces are identical, 4:0 corresponds to the opposite spectrum, two different identities. A threshold of 1.1 would classify every pair correctly.

6.2 Formation of Deep Convolutional Network

Considering our system as shown in the Figure 31, the Resnet model is treated as a blackbox and end-to-end learning is performed for which the triplet loss is employed.

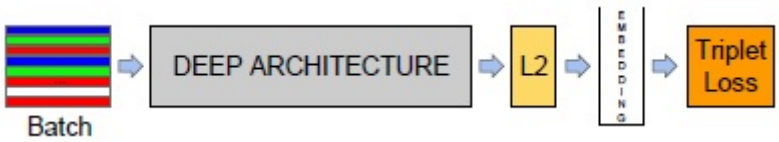

Figure 31 – FaceNet network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training

6.2.1 Triplet Loss

We strive for an embedding $f(x) \in \Re^d$, where x is an image, such that the squared distance between all faces, independent of imaging conditions, of the same identity is small,

whereas the squared distance between a pair of face images from different identities is large.

Here we want to ensure that an image x_i^a (anchor) of a specific person is closer to all other images x_i^p (positive) of the same person than it is to any image x_i^n (negative) of any other person. This is visualized in Figure 32.

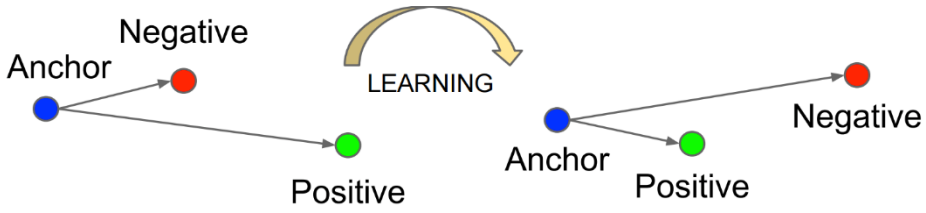

Figure 32 – The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity

The loss function that has to be minimized is:

$$L = \sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 + \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \tag{6.1}$$

where α is a margin that is enforced between positive and negative pairs. The set of all possible triplets in the training set has cardinality N.

6.2.2 Triplet Selection

It is crucial to select hard triplets, that are active and can therefore contribute to improving the model. This means that, given x_i^a, we want to select an x_i^p (hard positive) such that $argmax_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$ and similarly, x_i^n (hard negative) such that $argmin_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2$.

Selection of hard positive/negative exemplars is done from within mini-batches which are generated online. This is done as it is infeasible to find out the *argmin* and *argmax* of the entire training set all at once.

Selecting the hardest negatives can in practice lead to bad local minima early on in training, specifically it can result in a collapsed model (i.e. $f(x) = 0$). In order to mitigate this, it helps to select x_i^n such that

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2 \tag{6.2}$$

We call these negative exemplars semi-hard, as they are further away from the anchor than the positive exemplar, but still hard because the squared distance is close to the anchor-positive distance. Those negatives lie inside the margin .

6.2.3 Deep Convolution Network

In the implementation, the deep CNN is trained using Stochastic Gradient Descent (SGD) with standard backprop (13). In most experiments we start with a learning rate of 0.05 which we lower to finalize the model. The models are initialized from random and trained on a CPU cluster for 1,000 to 2,000 hours. The decrease in the loss (and increase in accuracy) slows down drastically after 500h of training, but additional training can still significantly improve performance. The margin α is set to 0.2. The Resnet architecture is used as the core Deep Convolutional network which is trained in the process (8) (see Figure 33). Instead of performing the training ourselves on our system, a pre-trained model provided by (14) has been used.

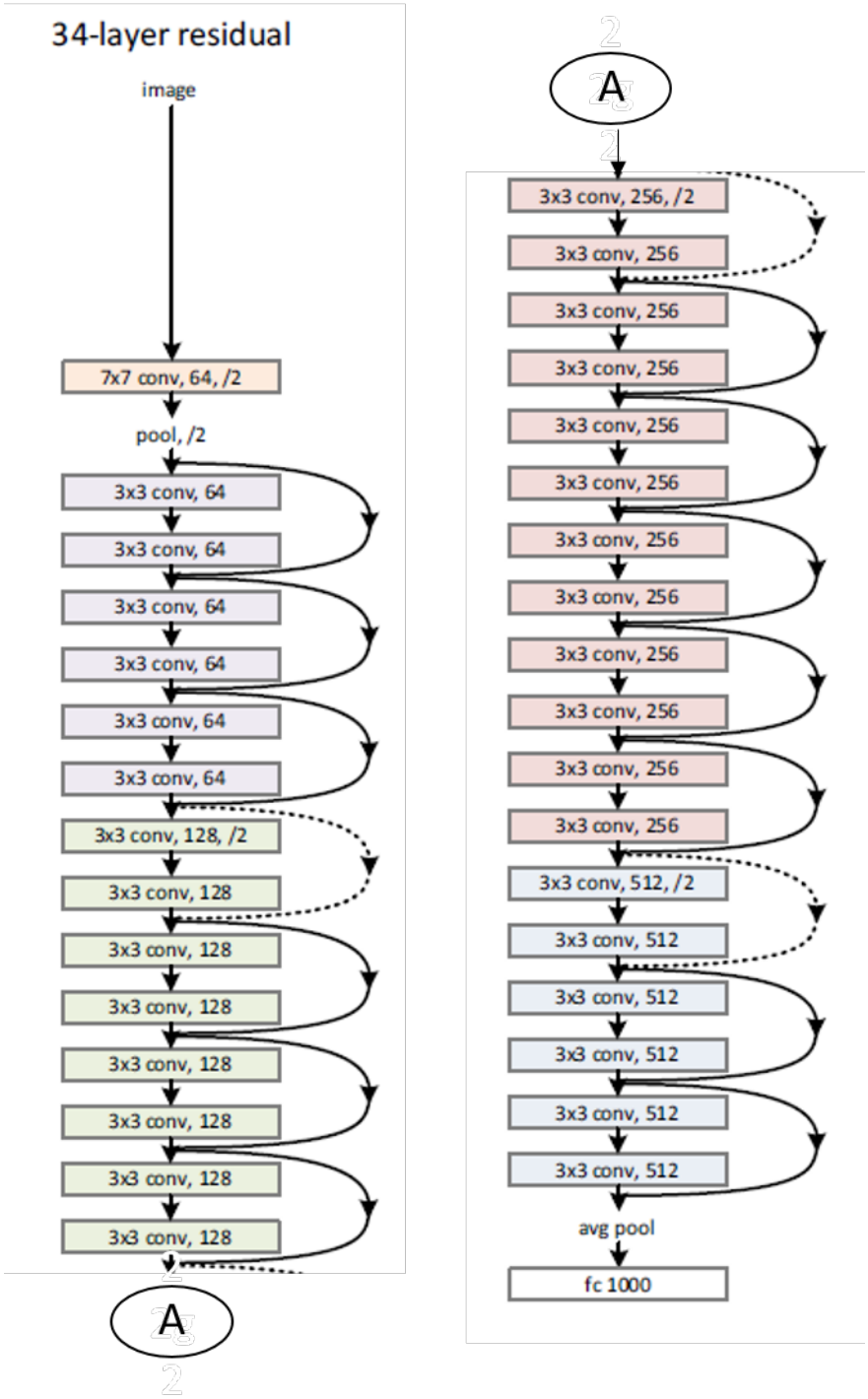

Figure 33 – A residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions.

7 Results

7.1 Face detection

Among the various algorithm available for face detection, we implemented Haar-features based face detection given by Viola-Jones and HOG features based face detection. Given below is the results of face detection when tested on number of images. We took images in which faces have different orientation, taken from different angles and in different illumination conditions. Some faces are not fully visible and some are occluded. These variations will give differentiated test results.

Figures 34, 35, 36 and 37 shows the result of face detection in which all detected faces are enclosed in a rectangle. In each of these figures, right side image is the result of Haar-features based detection and left side image is the result of HOG based detection.

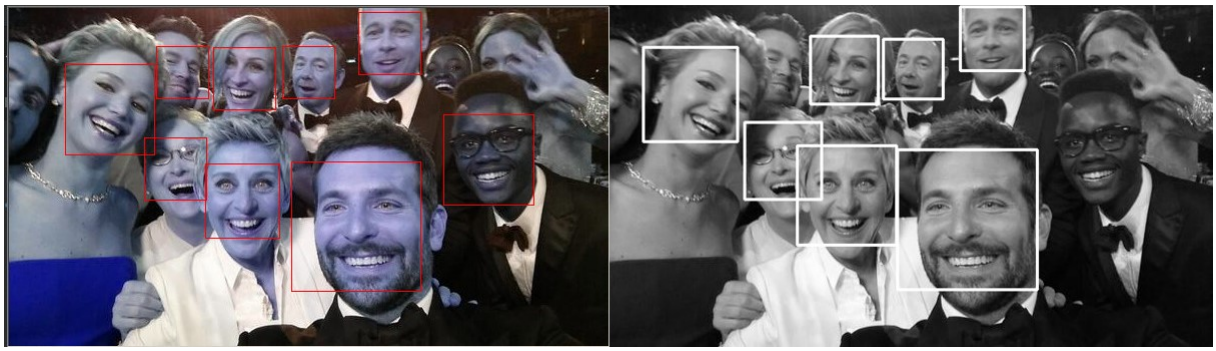

Figure 34 – Face detection test image 1

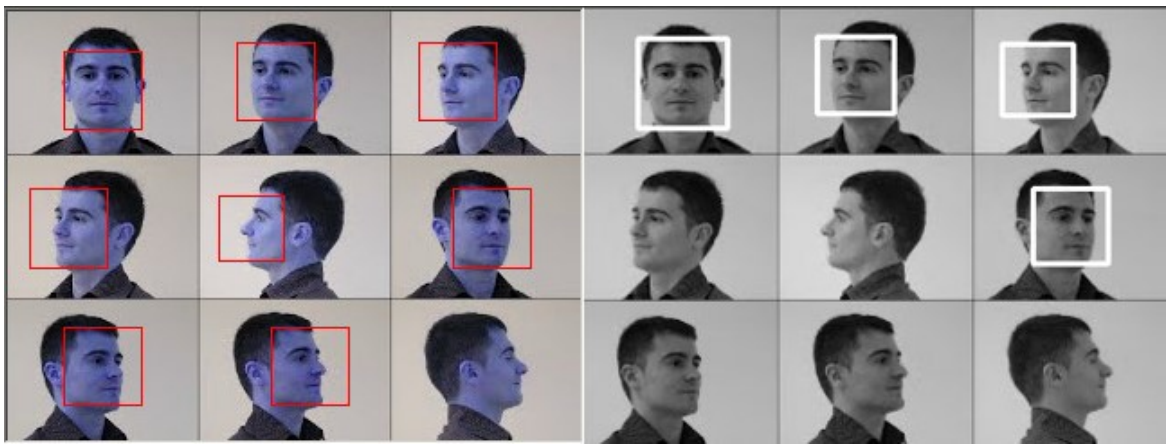

Figure 35 – Face detection test image 2

Figure 36 – Face detection test image 3

Figure 37 – Face detection test image 4

Table 3 and table 4 list the total number of faces in each image and number of detected faces with accuracy rate. From these results, we can infer that HOG based face detection is more accurate than Haar-feature based detection.

Image	Number of faces	Number of detected faces	Accuracy
Image 1	10	7	70.00%
Image 2	9	4	44.44%
Image 3	11	5	45.45%
Image 4	110	79	71.81%

Table 3 – Accuracy of Haar-features based Face detection

Image	Number of faces	Number of detected faces	Accuracy
Image 1	10	9	90.00%
Image 2	9	8	88.88%
Image 3	11	8	72.72%
Image 4	110	103	93.63%

Table 4 – Accuracy of HOG based Face detection

7.2 Face recognition

Figure 38 is the still image taken from video feed which shows the result of our face recognition program. Program detects all the faces in the current frame and label those faces which are already present in the database. Any face which is new to the system is labelled as "Unknown".

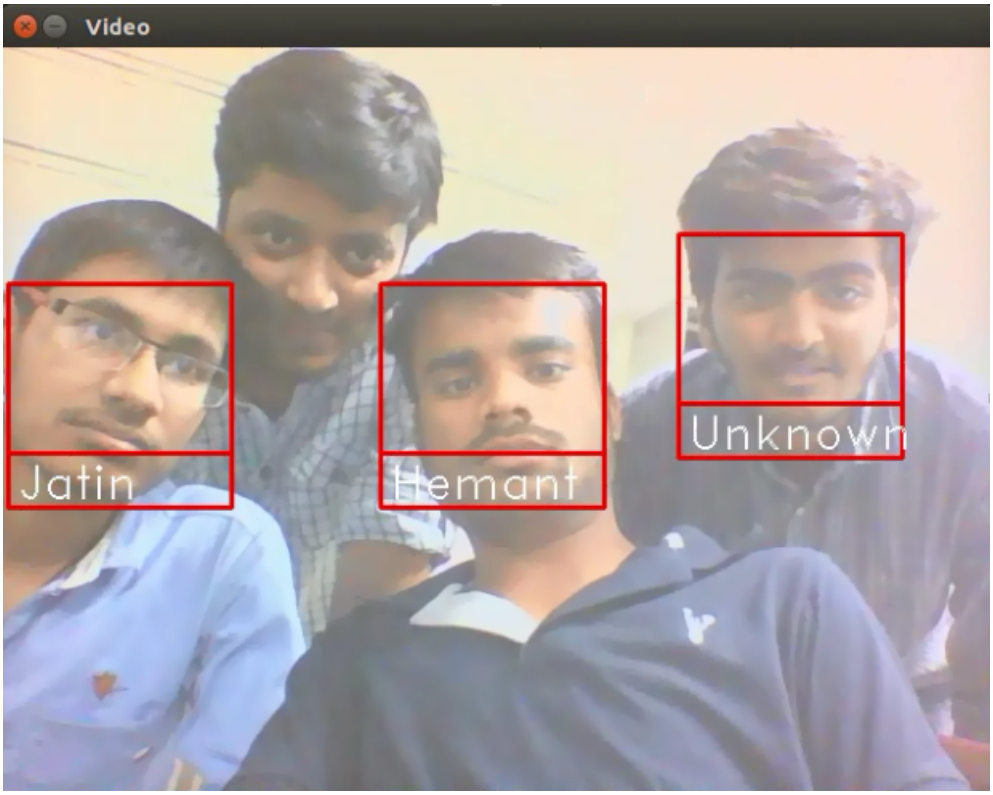

Figure 38 – Result of face recognition

Part IV

Conclusion

8 Conclusion

The project has been completed, where all the modules, namely – mechanical section, Audio-Visual Interaction System and Computer Vision Section have all achieved their individual tasks. A working prototype has been presented during the final evaluation of the project (2016-17) at the Department of Electrical Engineering, MNIT Jaipur.

The basic function of the Smart Animatronic Human Face have been achieved which includes Face tracking and recognition, Factoid Question Answering (Visual and Audio based). In future, we would want to work on integrating all the sections of the project which are now working separately. This would be used to achieve more nuanced objectives, such as automatic switching between different modes of operation of the Robotic Human Face, Real time interaction and Information Retrieval based on Audio and Visual signals.

The remainder of our project shall involve integrating all independently working modules into one coherent system with all the features intact and it responds to one's commands accordingly.

Bibliography

1 WESTON, J. et al. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

2 MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 3111–3119.

3 BORDES, A. et al. Artificial tasks for artificial intelligence. *Facebook AI Research. ICLR–San Diego–May*, v. 7, p. 2015, 2015.

4 SUKHBAATAR, S. et al. End-to-end memory networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 2440–2448.

5 IYYER, M. et al. A neural network for factoid question answering over paragraphs. In: *EMNLP*. [S.l.: s.n.], 2014. p. 633–644.

6 MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

7 TAIGMAN, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2014.

8 HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 770–778.

9 SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 815–823.

10 SUN, Y.; WANG, X.; TANG, X. Deeply learned face representations are sparse, selective, and robust. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 2892–2900.

11 SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 1–9.

12 WEINBERGER, K. Q.; BLITZER, J.; SAUL, L. K. Distance metric learning for large margin nearest neighbor classification. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2006. p. 1473–1480.

13 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Cognitive modeling*, v. 5, n. 3, p. 1, 1988.

14 DAVISKING. *Dlib Models*. [S.l.], Feb 2017 (accessed May 4, 2017). Disponível em: <https://github.com/davisking/dlib-models>.