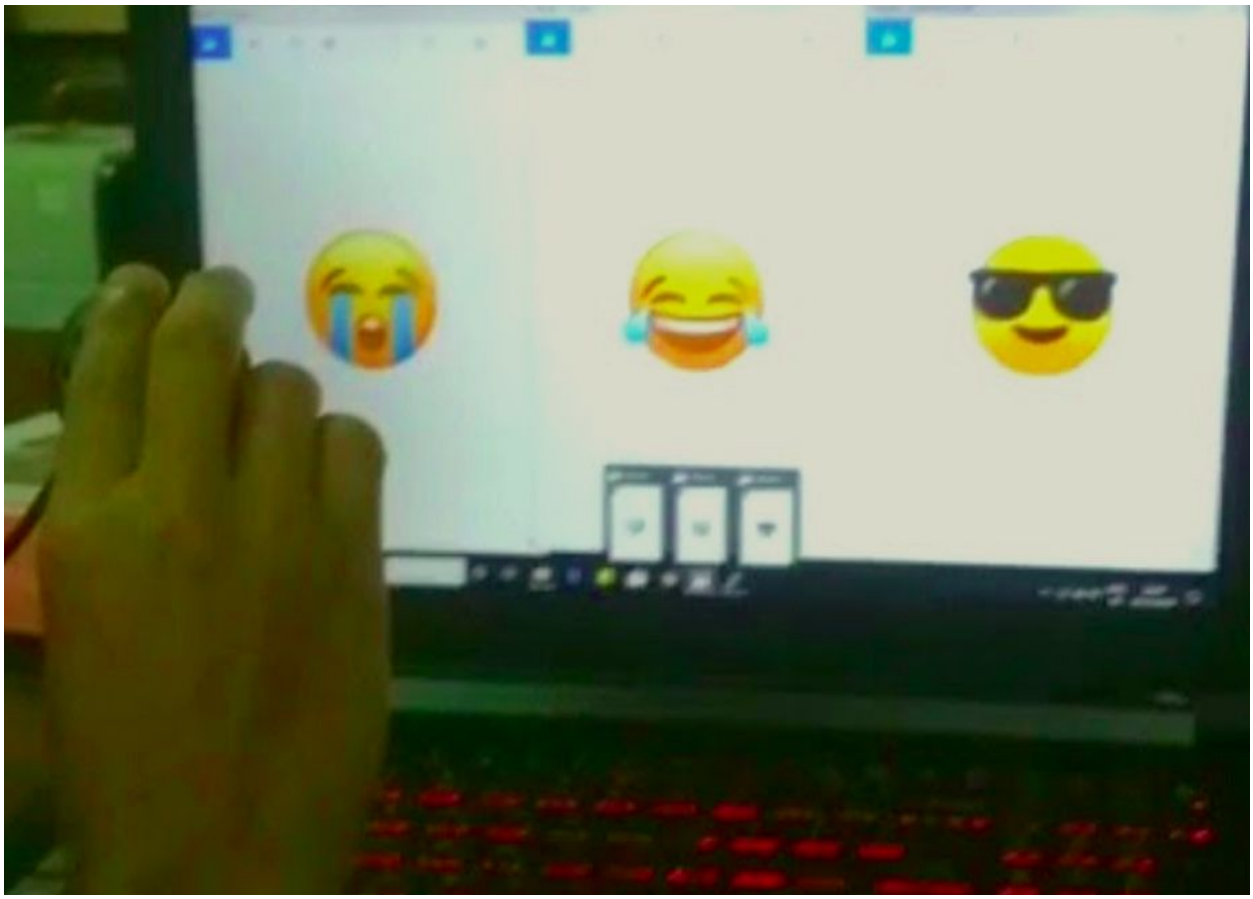
FPGA BASED EMOJI DETECTOR

Project Report
Team ID GP-69900151
Team Members: Vaibhav Malviya, Aryan Lall, Simranjeet Singh, Prashant Kurrey



Introduction

FPGAs have shown stronger potential for the new generation of machine learning algorithms and the Internet of Things applications. Its compute power, efficiency and ease-of-use is a major advantage of using FPGAs over other devices, thus creating a blend of Machine learning and IoT.
In this project we are trying to solve the real problem on a small scale thus we have used emoji as the different human faces with different emotions. We used one camera to capture the emoji and this data is further processed by the FPGA to detect the type of emoji.

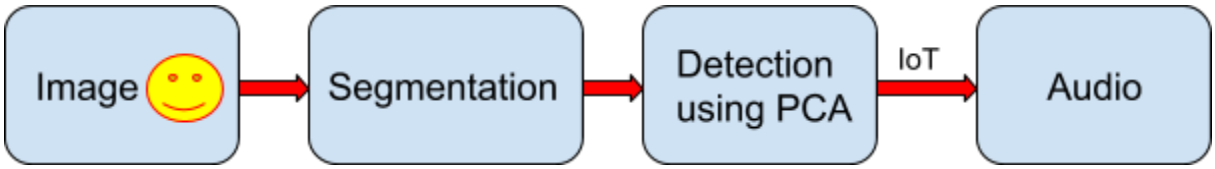
Motivation

We have seen the voice-controlled speaker, which plays the song when we call them. But what if someone is sad or depressed or something else and do not want to play the song. If we can detect human behavior then according to their behavior, a song or speech will be played.
This system can also be used as a safety purpose for the home, which can detect human body behavior and inform the owner.

Resources used

1. Pynq FPGA
2. Camera (Webcam)
3. IoT
4. Miscellaneous

System Flowchart

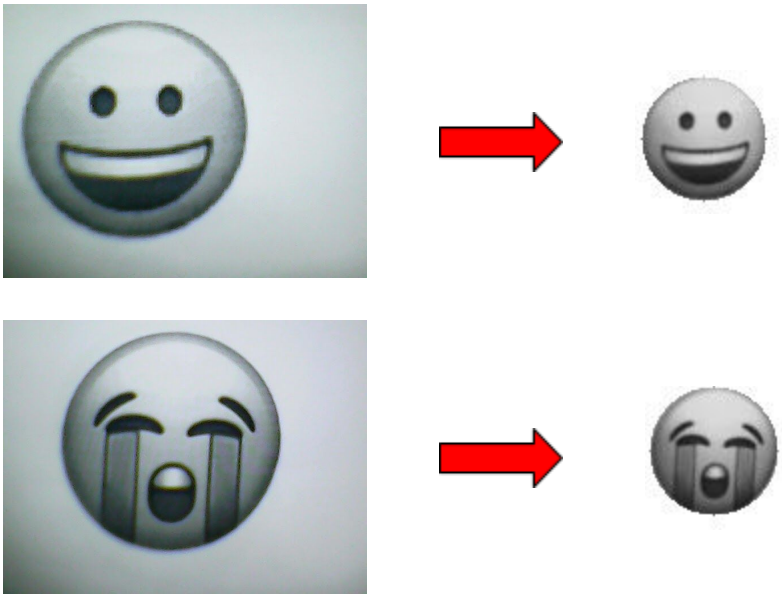


System Overview

1. Segmentation:

Segmentation is the first operation that we perform on the raw image. It is essential to remove the background noises or essentially making the background of the image completely white so as to work only with the emoji pixels.
Almost all emojis are circular, which makes it easier to be segmented. Since emoji could be placed anywhere in the image plane, we need to first bring it to the center (for the PCA technique to work efficiently, as explained later). After this, we perform segmentation using the **Hough circle** transform. The required parameters for the same are decided by sufficient trials.
Henceforth, we obtain a circular region which contains the emoji pixels. This region is cropped out of the image with the background being the white pixels. This segmented image is resized to a standard resolution of **72X72** pixels, thus making it uniform for all images to be detected.

Some samples for the working of segmentation are shown below:



These segmented images are further processed using the PCA technique, which detects the type of the emoji.

2. PCA (Principal Component Analysis):

We performed Principal Component Analysis Technique for Emoji detection. PCA technique reduces a large dimension of data to a smaller dimensionality in a feature space. We split the PCA technique into 2 halves. The first one is the training of the dataset and the next one is testing the captured image.
The testing part was performed in Matlab where the resulting data was saved in the PYNQ board which was used later for the testing phase.

ALGORITHM:

1. *Training:* We collected a dataset of emojis with various emotions. The emojis considered were: **Crying, Unwell, Concerned, Glasses (SWAG), Laugh and Smile**. 16 different images for each of the emoji were taken in for training purposes. The images array was flattened and stacked over one another. Then the actual matrix operations were carried on. Step by Step technique:
    a. The covariance matrix is calculated for the train images. It tells how one variable is associated with another is quite powerful.
    b. Next, eigenvectors and eigenvalue are calculated. Eigenvectors represent directions. And eigenvalues represent in importance in terms of magnitude corresponding to a particular vector.

c. After eigendecomposition, the Eigen coefficients are calculated for each of the trained imaged using the eigenvectors and the original flattened images array.
d. Later this eigenvector and Eigen coefficient and saved on the PYNQ board which are used for testing images.

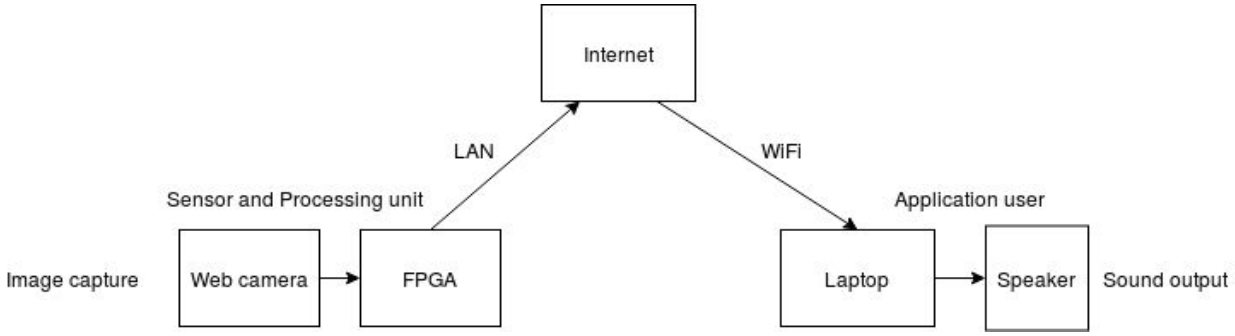
2. *Testing:* The Eigen coefficients and vector is used to test the captured image.
    a. The captured image is again first flattened and its corresponding Eigen coefficient is calculated.
    b. The found eigen coefficient is compared with the trained images eigen coefficient and the closest image with the lowest mean square error is calculated.
    c. The image with minimum error is the image with which the test image resembles.

IMPLEMENTATION:

In our work, we capture images from a webcam, passed over our PCA algorithm which detects the emotion. To make it real-time, webcam captures images continuously and find the emotion on each image. After the image is captured, we find the emotion which has occurred a maximum number of times. Thus the emotion of the test image is detected, which is sent over wifi to some other PC via socket connection over Wifi using the Internet Of Things (IoT).

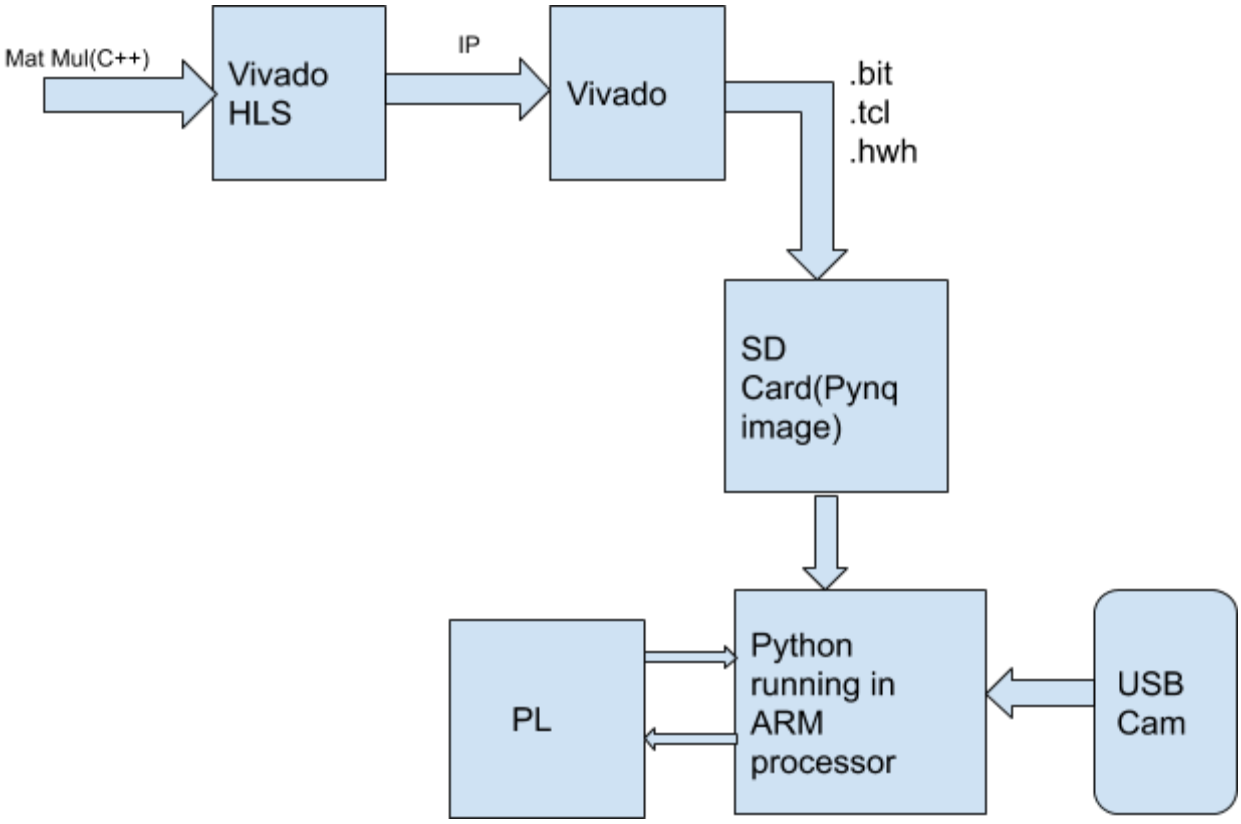
3. Internet of Things:

We have implemented IoT using FPGA plus camera as the sensor and computation unit, and a laptop as the application user. The FPGA is connected to the internet through LAN which will act as a gateway. This connection is only one to one socket connection. The FPGA creates the server with port Id XYZ. After identifying the behavior of the emoji, the behavior id will be sent through the server created by FPGA. The second laptop will receive the behavior id through the gateway, depending on the behavior id, a song will be played on the laptop stored on the laptop.



FPGA v/s ARM Processor

Image captured using the USB device is done in the ARM processor and our algorithm uses extensive matrix multiplication with float values. We decided to move the matrix multiplication part to FPGA for a better rate. We used the Vivado HLS to implement the matrix multiplication IP. The IP generated by the HLS is then included in the Vivado along with ZYNQ SoC of Pynq z2.
Vivado generates the Bitstream, tcl, and Hwh that are further being used in the python. Refer to the following diagram for more information.



There are three important files used by the python code to program the PL.

1. ***.bit:** This the actual bitstream that contains the logic implementation. Using the pynq Overlay library this bitstream will be written into the PL.
2. ***.tcl(Tool command language):** This file has the information of block and wiring used in the design. Like which wire of IP is connected to which port of ZYNQ, name of custom IP imported from HLS, etc.
3. ***.hwh:** The ARM core and PL are memory-mapped. This file contains all the memory locations for interfacing. To access the PL we have to write on the particular address and will get data back on a particular memory address. This file keeps all the addresses for use.

Hardware v/s Software Task:

Software	Hardware
Image Capture	Segmentation
OpenCv reshaping	PCA Testing Algorithm
Socket Encoding	

Timeline:

- *First Week:* Since we all were new to FPGA background we familiarize ourselves with an introduction to PYNQ FPGA and its working.
- *Second Week:* Since we need to write the code in python and the board was not available to us we started writing basic code in python on PC. Parallelly database searching for emojis was going on.
- *Third Week:* Once the dataset was finalized the Training was done in Matlab and a very rough was written into the PC to test the algorithm. A tutorial regarding the AXI protocol and Vivado Design suite was started.
- *Fourth Week:* Once we got the board we started working on it. **Since we expected the Pynq Z-2 board but received Ultra96 board, our team flow was hampered**. However we can install the pynq based image in the ultra96 but we stick to pynq board because to work on ulta96 the bit stream file needs to be changed. Thus we in our institute talked to the professor and got the Pynq Z2 board issued.

- *Fifth-Sixth Week:* We understood the working of Pynq Board and Installed its image. Connected the board to the LAN connection and uploaded the codes into it. Started the training and testing by capturing the images from the webcam. In the meantime we started working on Vivado HLS.
- *Final Weeks:* We generated the bitstream from the hardware part and did the final testing part on the Pynq Board.

Results:

We tried testing out data on approximately 36 images…

EMOTION	No. of Test Images	Correctly Detected
Cry	6	6
Laugh	6	6
Concerned	6	4
Glasses (SWAG)	6	6
Smile	6	5
Unwell	6	5

Thus a total of 32 images were correctly detected as compared to 36 test images. Corresponds to 88.88% accurate train data.
The PCA approach thus provides a practical solution that is well fitted to the problem of face recognition. It is fast, relatively simple and has been shown to work well in a constrained environment.

Challenges Experienced:

- **Dataset:**
  - Getting the required dataset
  - Different emoji meaning everywhere and need to compile data from various websites
- **Segmentation:**
  - In the images captured by webcam, the emoji could be present anywhere in the image plane, which poses a challenge. It has to be first brought to the center for efficient detection. We have accounted for this.

- The noisy background of the emoji could lead to the wrong detection. Thus the emoji has to be cropped out from the image with a noise-free white background (as shown in the images in previous sections).

Future Improvement and Scope

- This system can be implemented using human faces which have varieties of behavior and furthermore this can be extended to the human body behavior
- A neural network can be made on this to increase the precision of the output
- This system can be implemented on WiFi connected speaker which can play the song according to the behavior of the human
- This system can be used for the surveillance of the home for safety purpose