IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Gesture Recognition Using TinyML Devices

Home Automation Applications

[1]Dheeraj Perumandla, [2] Ayisha Ryhana D, [3]Manognya Pendyala

[123]B. Tech Student

[1]Department of Information Technology, [2]Department of Computer Science, [3]Department of Mechanical Engineering,

[13]Sreenidhi Institute of Science and Technology, Hyderabad, India

[2]National Institute of Technology Puducherry, Puducherry, India

Abstract: Automating things has always been a part of development; it helps to make man's life simpler, and in a few cases, lives of physically challenged people a lot simpler. The use of gesture recognition is one of the ways to automate things. Gesture recognition can be done in many ways, and the use of accelerometer and gyroscope sensors is one way to do it efficiently. Home automation is one of the use cases of gesture recognition, and after recognizing the gesture, each gesture is programmed to do a specific task on home appliances.

Index Terms **- Deep Learning, Gesture Recognition, IoT Applications, Home Automation, Accelerometer, Gyroscope, Neural Networks.**

I. INTRODUCTION

A gesture is the movement of body parts to communicate an idea or message. Hand gestures are a common way of communication. Gesture recognition aims to interpret these human gestures via mathematical algorithms. This can help bridge a communication gap by translating the sign language or enable users to control or interact with devices without physically touching them. It is also possible for gesture recognition to replace conventional input methods in the majority of our everyday devices.

We have worked on mapping gestures to perform various activities, for this purpose, the machine should be intelligent enough to identify the gesture in the first place and later that identified gesture can trigger an activity to perform. The applications of gesture recognition are vast and home automation is one of them, this home automation can be made more useful for physically challenged and could turn their lives a little less harder.

II. METHODS OF RECOGNITION

An important factor of gesture recognition is the method of gesture recognition. Keeping in mind the home automation application, the following methods were considered.

1. Hand pose estimation using cameras [1]

While pose estimation is a promising technique in gesture recognition, the use of cameras can pose a few problems. While attempting to control the appliances of an entire house using pose estimation, there are two ways of camera placement. One, placing the camera in a fixed position with limited coverage. This is inconvenient as we will have to position ourselves in the camera's field of view to perform a gesture each time. Another option is to place multiple cameras to cover every angle. This is a possible security risk and invasion of privacy.

2. Hand pose estimation using IR tracking [2]

Gestures are mapped by the movement of the hand in the field of the InfraRed projection. Since the projection angle and area are fixed, we would have to move into its field of projection like with the cameras. This is again an inconvenience.

3. Using the accelerometer and gyroscope sensors

These sensors return accelerometer and gyroscope[5][6] values along the x, y, and z axes recorded during the movement of a hand to form a gesture. Embedding these sensors in a wearable device simplifies gesture recognition. This allows gestures to be recorded from any location within the house and does not require constant supervision of the entire house.

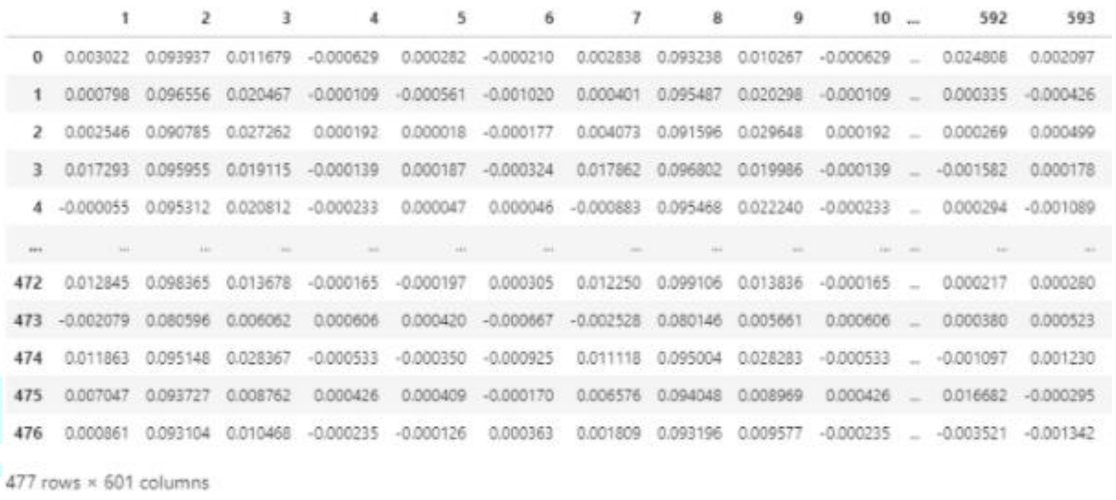
III. Dataset Collection And Transformation

Hand gestures are mapped to the corresponding sensor values recorded during the motion of the hand, i.e., accelerometer and gyroscope sensor values along x, y, and z axes. These 6 values were recorded 100 times for each gesture, i.e., 600 data points for one gesture.

In the real world implementation, the dataset would be recorded using Arduino, raspberry pi, etc. like devices. But due to the unavailability of such a device at hand, the sensors in the smartphones were used as an alternative.

An android app was written to read the values and write to a text file along with the corresponding gesture name. Each gesture was one line of the file with 601 values (600 sensor values and one gesture name). The text files were then consolidated using the CSV library in python.

The dataset was then normalized using the sklearn library, each sample independently (along axis 1). It has around 60 instances each of 8 recorded gestures – down-to-up, forward-clockwise, left-fall, up-clockwise, up-anticlockwise, left-to-right, right-to-left, forward-fall.



	1	2	3	4	5	6	7	8	9	10	...	592	593
0	0.003022	0.093937	0.011679	-0.000629	0.000282	-0.000210	0.002838	0.093238	0.010267	-0.000629	...	0.024808	0.002097
1	0.000798	0.096556	0.020467	-0.000109	-0.000561	-0.001020	0.000401	0.095487	0.020298	-0.000109	...	0.000335	-0.000426
2	0.002546	0.090785	0.027262	0.000192	0.000018	-0.000177	0.004073	0.091596	0.029648	0.000192	...	0.000269	0.000499
3	0.017293	0.095955	0.019115	-0.000139	0.000187	-0.000324	0.017862	0.096802	0.019986	-0.000139	...	-0.001582	0.000178
4	-0.000055	0.095312	0.020812	-0.000233	0.000047	0.000046	-0.000883	0.095468	0.022240	-0.000233	...	0.000294	-0.001089
...	...	...	...	...	...	...	...	...	...	...	...	...	...
472	0.012845	0.098365	0.013678	-0.000165	-0.000197	0.000305	0.012250	0.099106	0.013836	-0.000165	...	0.000217	0.000280
473	-0.002079	0.080596	0.006062	0.000606	0.000420	-0.000667	-0.002528	0.080146	0.005661	0.000606	...	0.000380	0.000523
474	0.011863	0.095148	0.028367	-0.000533	-0.000350	-0.000925	0.011118	0.095004	0.028283	-0.000533	...	-0.001097	0.001230
475	0.007047	0.093727	0.008762	0.000426	0.000409	-0.000170	0.006576	0.094048	0.008969	0.000426	...	0.016682	-0.000295
476	0.000861	0.093104	0.010468	-0.000235	-0.000126	0.000363	0.001809	0.093196	0.009577	-0.000235	...	-0.003521	-0.001342

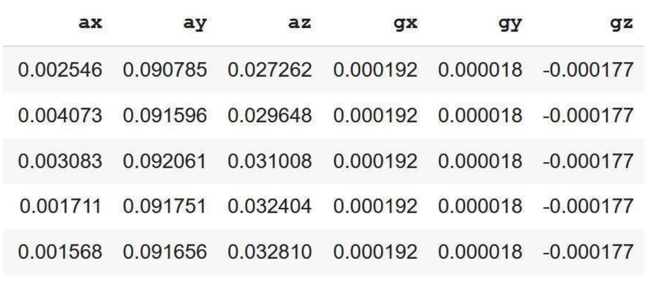
477 rows × 601 columns

Figure 1. Dataset

This is the dataset with 477 rows and 601 columns.

IV. Visualization

The recorded data set is copied into a new variable, and few manipulations are made on the copied data in order to make it ready for visualization. The modified data set looks like the table below.



ax	ay	az	gx	gy	gz
0.002546	0.090785	0.027262	0.000192	0.000018	-0.000177
0.004073	0.091596	0.029648	0.000192	0.000018	-0.000177
0.003083	0.092061	0.031008	0.000192	0.000018	-0.000177
0.001711	0.091751	0.032404	0.000192	0.000018	-0.000177
0.001568	0.091656	0.032810	0.000192	0.000018	-0.000177

Figure 2. Modified dataset for visualization purpose

The data was separated for each gesture and visualization of the accelerometer and gyroscope data points is done separately for each gesture, respectively. Visualization was done with the help of `matplotlib.pyplot`

1. **For Gyroscope axes points:**

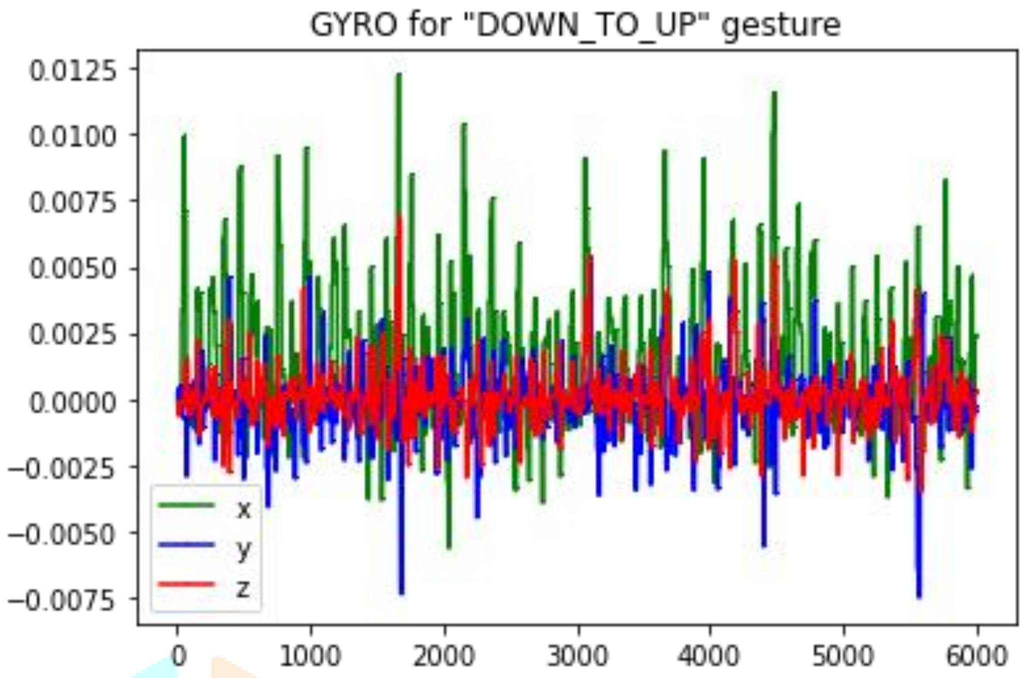


Figure 3. Visualization of gyroscope points of ‘Down-to-up' gesture.

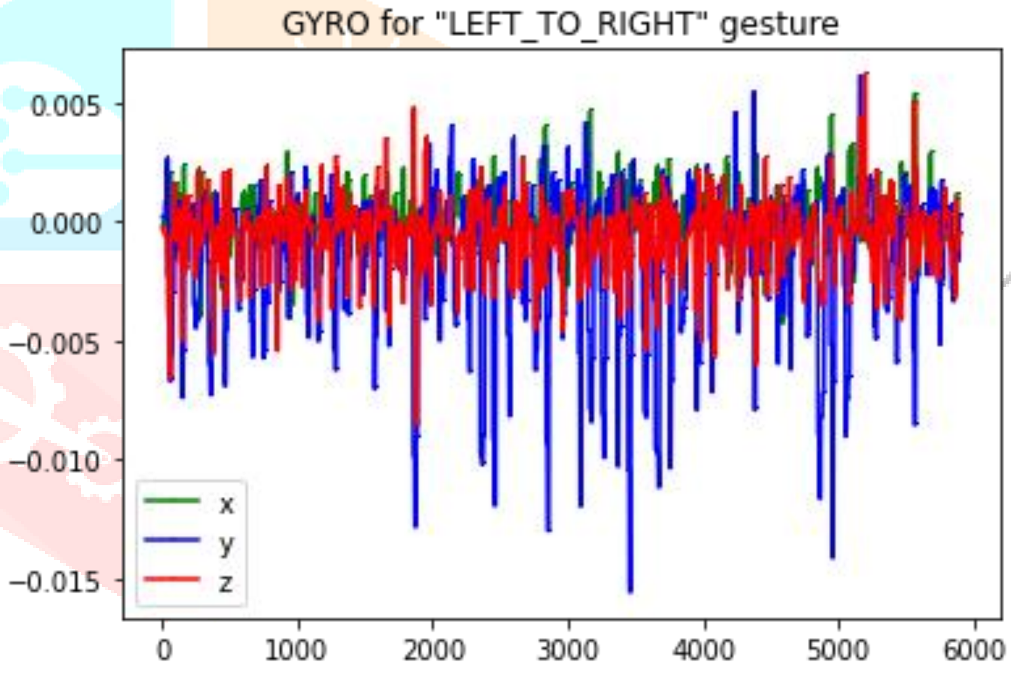


Figure 4. Visualization of gyroscope points of 'Left-to-right' gesture.

2. For Accelerometer axes points:

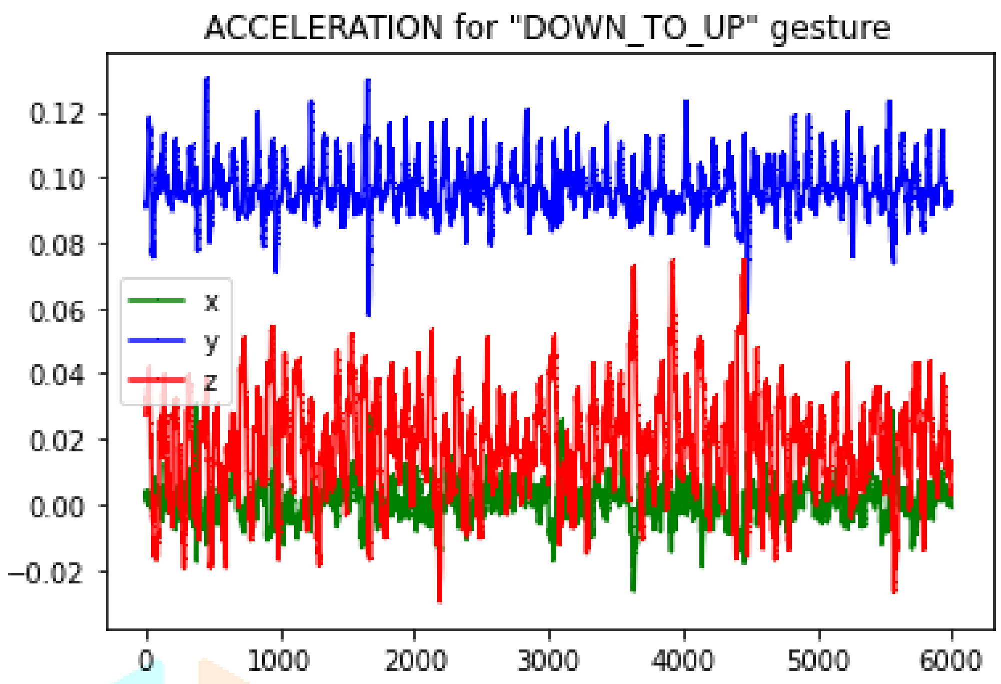


Figure 5. Visualization of accelerometer points of 'Down-to-up' gesture.

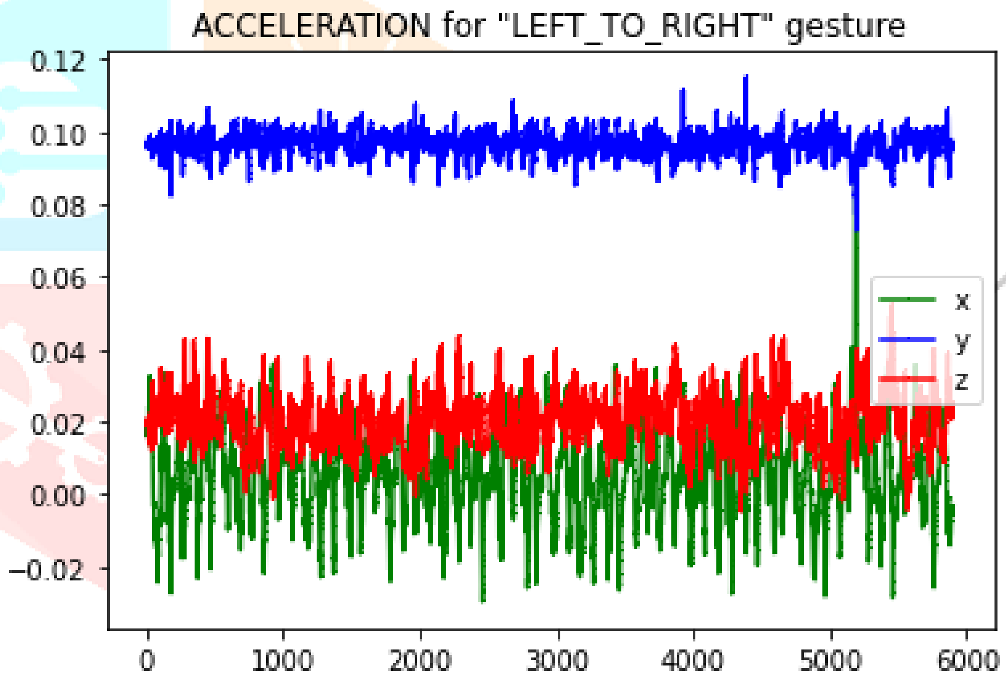


Figure 6. Visualization of gyroscope points of 'Left-to-right' gesture.

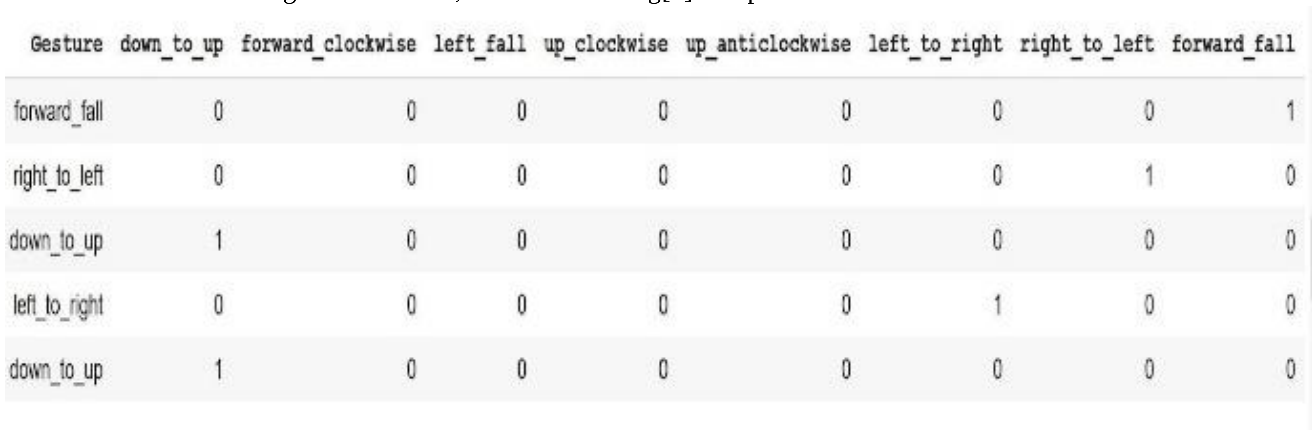
After visualizing, the data was analysed for any irregularities or outliers. One was found in the 'left_to_right' acceleration plot and removed.

V. BUILDING A DEEP NEURAL NETWORK.

After performing EDA on data, the data is now ready for feeding it to a neural network.

1. Categorical class labels to numerical.

As the class labels were categorical in nature, One Hot encoding[7] was performed to convert them to numerical data.



Gesture	down_to_up	forward_clockwise	left_fall	up_clockwise	up_anticlockwise	left_to_right	right_to_left	forward_fall
forward_fall	0	0	0	0	0	0	0	1
right_to_left	0	0	0	0	0	0	1	0
down_to_up	1	0	0	0	0	0	0	0
left_to_right	0	0	0	0	0	1	0	0
down_to_up	1	0	0	0	0	0	0	0

Figure 7. Features after One Hot encoding.

Now, the data is suitable to feed it to the neural network.

2. Creating a neural network model.

Now a sequential neural model is initialized and later 16 dense neurons were added to the 1st hidden layer with activation as 'relu' succeeding the input layer of size 600 in our case. Now two hidden layers are added later to the 1st hidden layer with 8 and 4 neurons to each layer in stage with activation as 'relu'. Now a final layer is added with a number of class labels as a number of neurons with activation as 'softmax'.

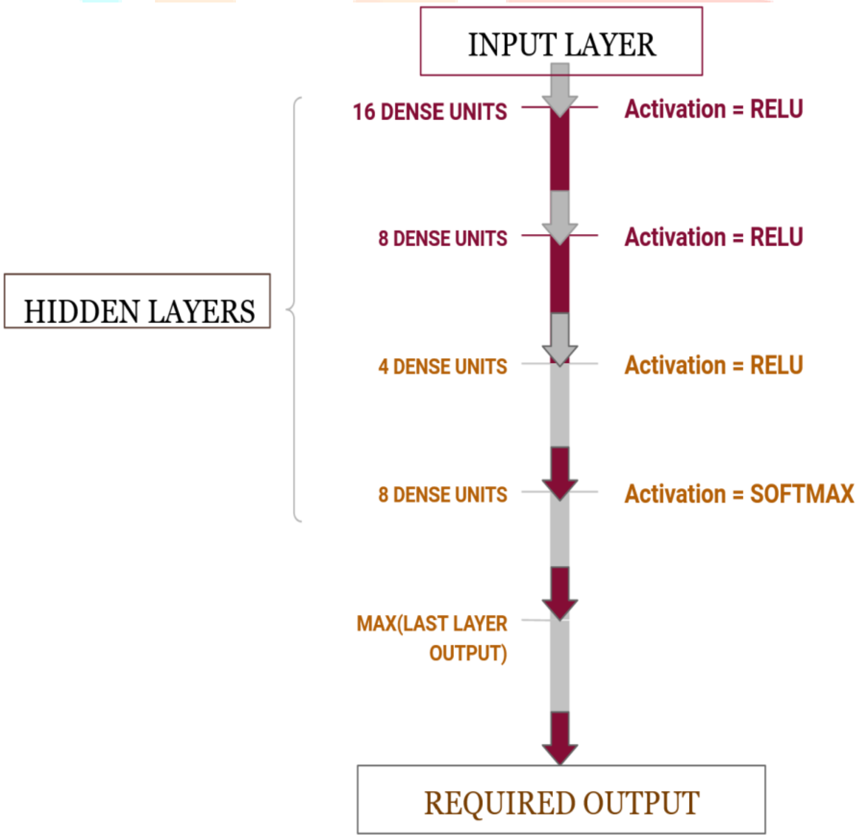
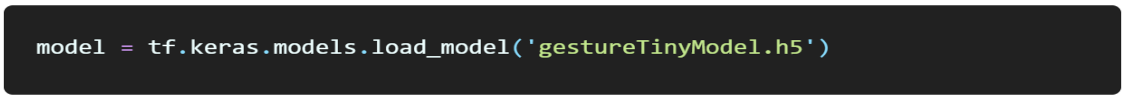


Figure 8. Flow chart of the neural network.

VI. TRAINING AND TESTING

After the neural network is built, the training data is fit on the model with epoch equal to 256 and with batch_size equal to 32. Later, predictions on test data were found by using the trained neural network model for which accuracy of 98.9% with a loss of 0.2 is achieved. After achieving a reliable accuracy, the model is saved using '.h5' extension.



```
model = tf.keras.models.load_model('gestureTinyModel.h5')
```

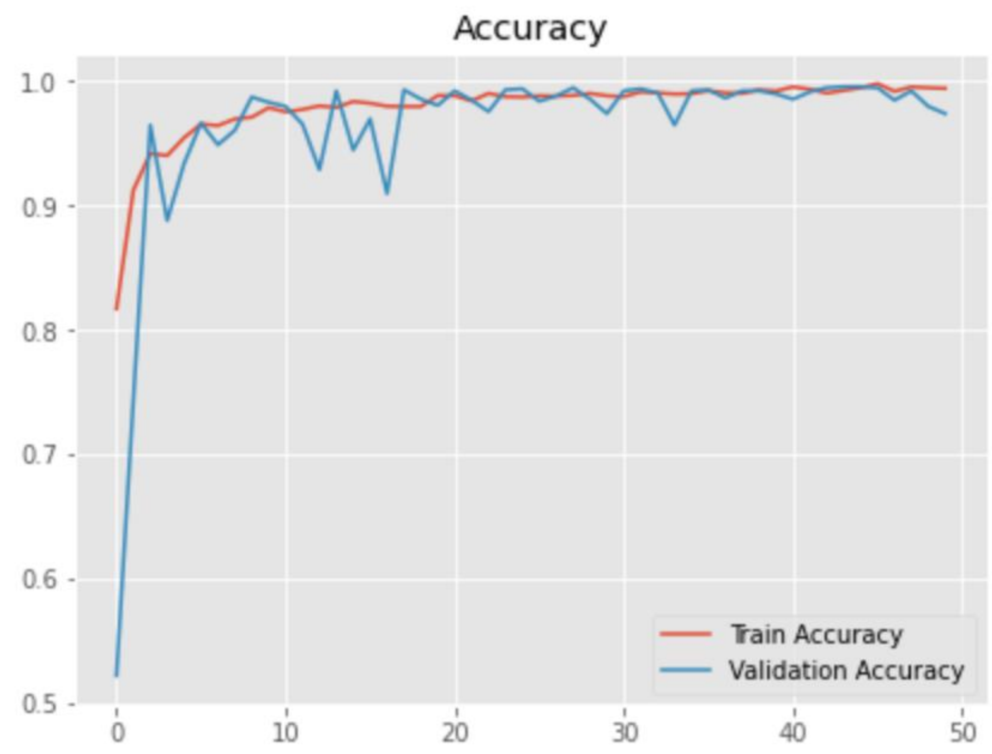


Figure 9. No.of Epochs vs Accuracy for both Training and Validation

Figure 9 depicts the comparison between Training and Validation accuracy, when Accuracy is mapped against number of epochs.

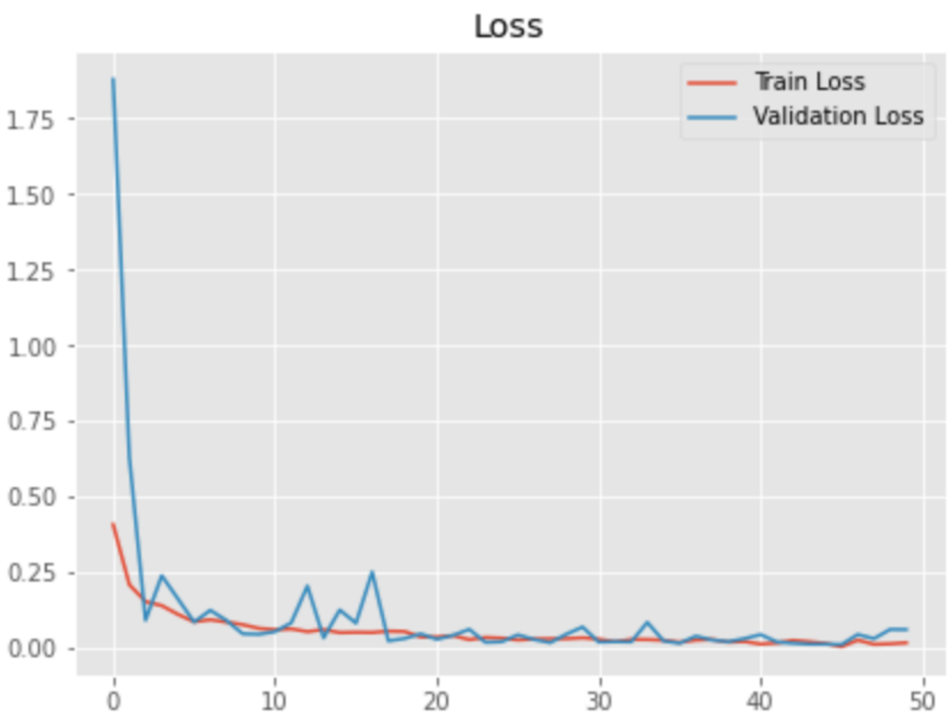


Figure 10. No.of Epochs vs Loss for both Training and Validation

Figure 10 depicts the comparison between Training and Validation loss, when loss is mapped against number of epochs. From the above two graphs it is understood that there is no over fitting.

VII. Using deepC[3]

1. Info and design of deepC[3]

deepC is a deep learning compiler and inference framework to perform deep learning in edge devices like raspberry pi, Arduino, node MCU, etc. deepC uses ONNX[4] as a frontend, which enables the compilation of deep learning models from popular frameworks like TensorFlow, PyTorch, Keras. In the backend, deepC[3] uses the LLVM compiler toolchain[8], which helps to produce optimized executables for resource-constrained devices.

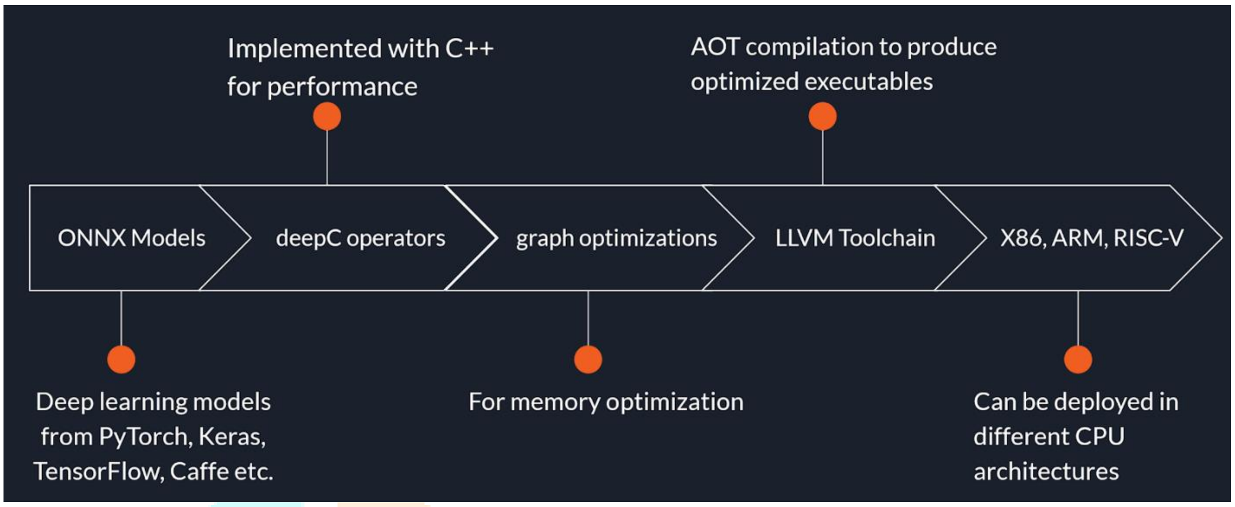

Implemented with C++ for performance
AOT compilation to produce optimized executables
ONNX Models
deepC operators
graph optimizations
LLVM Toolchain
X86, ARM, RISC-V
Deep learning models from PyTorch, Keras, TensorFlow, Caffe etc.
For memory optimization
Can be deployed in different CPU architectures

Figure 11. Design of deepC

2. Compilation using deepC

The saved model is compiled using the deepC compiler which gives us in machine code which can be dumped into resource-constrained devices like Arduino. `!deepCC gestureTinyModel.h5` command is used to compile the saved model.
Now, when the model is running on an edge device, it recognizes the trained gestures, which can be further programmed to perform certain tasks per each gesture.

VIII. Gestures Used In This Project And The Tasks Mapped To Them.

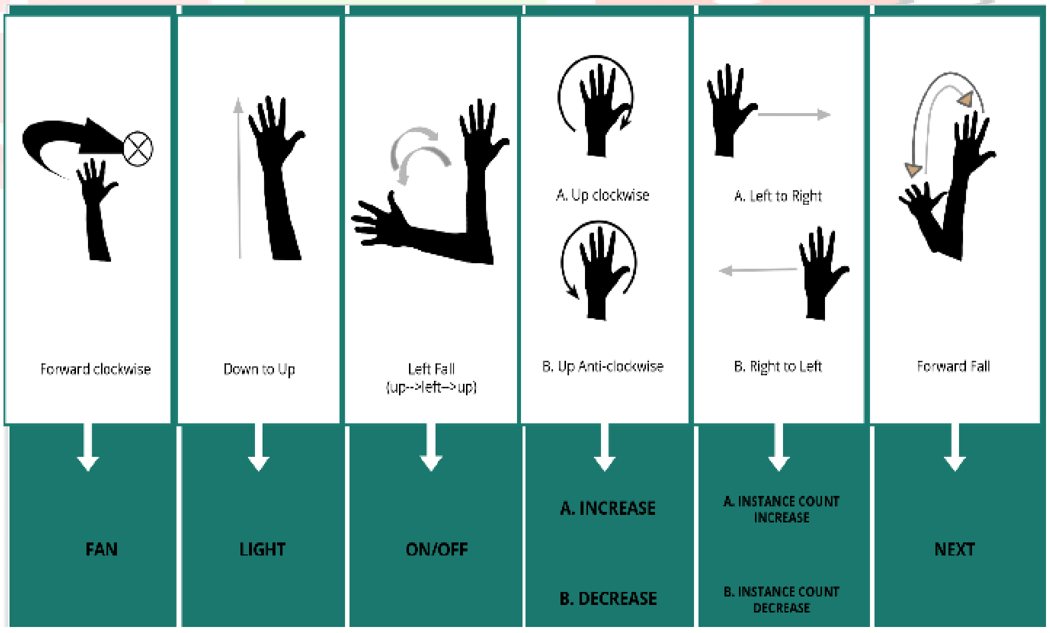


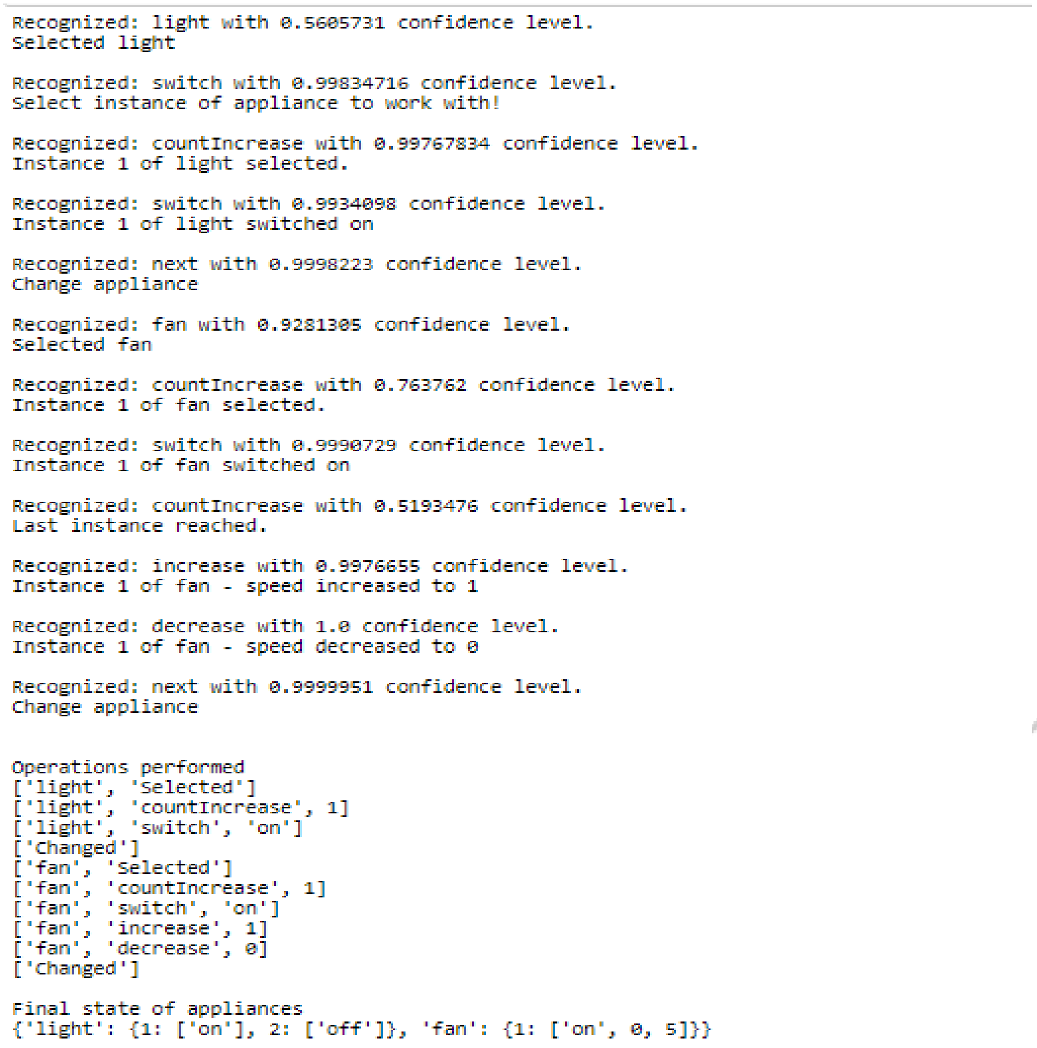
Figure 12. Tasks mapping with gestures.

In our experiment, Forward Clockwise gesture is mapped to select Fan as the appliance and Down to Up gesture is mapped to select Light as the appliance. Now, Left Fall i.e. up to Left to Up gesture is mapped to on/off the selected appliance. Up Clockwise is to increase any parameter of a particular appliance, for example, to increase speed of Fan. Up Anti-clockwise to decrease any parameter of a selected appliance, for example, to decrease the speed of Fan. Left to right gesture maps to instance count increase for example, Fan-1 to Fan-2 and right to left gesture maps decrease in instance count for example, Fan-2 to Fan-1 and Forward Fall gesture maps to next scenario.

IX. Home Automation Application

This is a POC for gesture-based home automation system - controlling the various appliances at home with hand gestures. The aim is to embed the sensors in a wearable device. We move our hands in so many different ways, and there is a high chance that one of these many movements map to the recorded gestures and trigger responses. To counter this, we can include a tiny button/other mechanisms to indicate the start of the gesture.

Each appliance type, the set of operations for the appliances and a few other operations necessary for the home automation system (moving between instances of an appliance, like fan1 to fan2 or fan2 to fan1, or switching between appliance types) are mapped to corresponding gestures. A list of appliances installed at home and the status of the individual instances are maintained.
Each gesture performed is recognized, and its validity in reference to the previous gestures is checked. If it is valid, the operation is performed and logged.



```
Recognized: light with 0.5605731 confidence level.
Selected light

Recognized: switch with 0.99834716 confidence level.
Select instance of appliance to work with!

Recognized: countIncrease with 0.99767834 confidence level.
Instance 1 of light selected.

Recognized: switch with 0.9934098 confidence level.
Instance 1 of light switched on

Recognized: next with 0.9998223 confidence level.
Change appliance

Recognized: fan with 0.9281305 confidence level.
Selected fan

Recognized: countIncrease with 0.763762 confidence level.
Instance 1 of fan selected.

Recognized: switch with 0.9990729 confidence level.
Instance 1 of fan switched on

Recognized: countIncrease with 0.5193476 confidence level.
Last instance reached.

Recognized: increase with 0.9976655 confidence level.
Instance 1 of fan - speed increased to 1

Recognized: decrease with 1.0 confidence level.
Instance 1 of fan - speed decreased to 0

Recognized: next with 0.9999951 confidence level.
Change appliance


Operations performed
['light', 'Selected']
['light', 'countIncrease', 1]
['light', 'switch', 'on']
['Changed']
['fan', 'Selected']
['fan', 'countIncrease', 1]
['fan', 'switch', 'on']
['fan', 'increase', 1]
['fan', 'decrease', 0]
['Changed']

Final state of appliances
{'light': {1: ['on'], 2: ['off']}, 'fan': {1: ['on', 0, 5]}}
```

Figure 13. This is a sample of the sequence of operations performed using the gesture recognition system and the subsequent home automation logic.

X. References

[1] Erol, Ali, et al. "Vision-based hand pose estimation: A review." *Computer Vision and Image Understanding* 108.1-2 (2007): 52-73.
[2] Rohith, H. R., Shiva Gowtham, and A. S. Sharath Chandra. "Hand gesture recognition in real time using IR sensor." *International Journal of Pure and Applied Mathematics* 114.7 (2017): 111-121.
[3] Rohit Sharma et. Al, DNNC: Deep Neural NetworkCompiler,2019 https://github.com/ai-techsystems/deepC
[4] Jin, Tian, et al. "Compiling ONNX Neural Network Models Using MLIR." arXiv preprint arXiv:2008.08272 (2020).
[5] Hyde, Rick A., et al. "Estimation of upper-limb orientation based on accelerometer and gyroscope measurements." IEEE Transactions on Biomedical Engineering 55.2 (2008): 746-754.
[6] Ha, Sojeong, and Seungjin Choi. "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors." 2016 International Joint Conference on Neural Networks (IJCNN). IEEE, 2016.
[7] Seger, Cedric. "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing." (2018).
[8] Rinta-Aho, Teemu, Mika Karlstedt, and Madhav P. Desai. "The click2netfpga toolchain." 2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12). 2012.