iMirror: A Smart Mirror for Stress Detection in the IoMT Framework for Advancements in Smart Cities

L. Rachakonda[1], Pritivi Rajkumar[2], S. P. Mohanty[3], E. Kougianos[4]

University of North Texas, Denton, TX 76203, USA.[1,2,3,4]

Email: rachakondalaavanya@my.unt.edu[1], pritivirajkumar@my.unt.edu[2], saraju.mohanty@unt.edu[3] and elias.kougianos@unt.edu[4]

Outline Of Talk

- Introduction
- Motivation
- Importance of Stress
- Existing Solutions - their Issues
- Proposed Solution
- Novel Contributions
- Architecture of iMirror
- Proposed Methodology of iMirror
- Implementation of iMirror
- Conclusions and Future Research

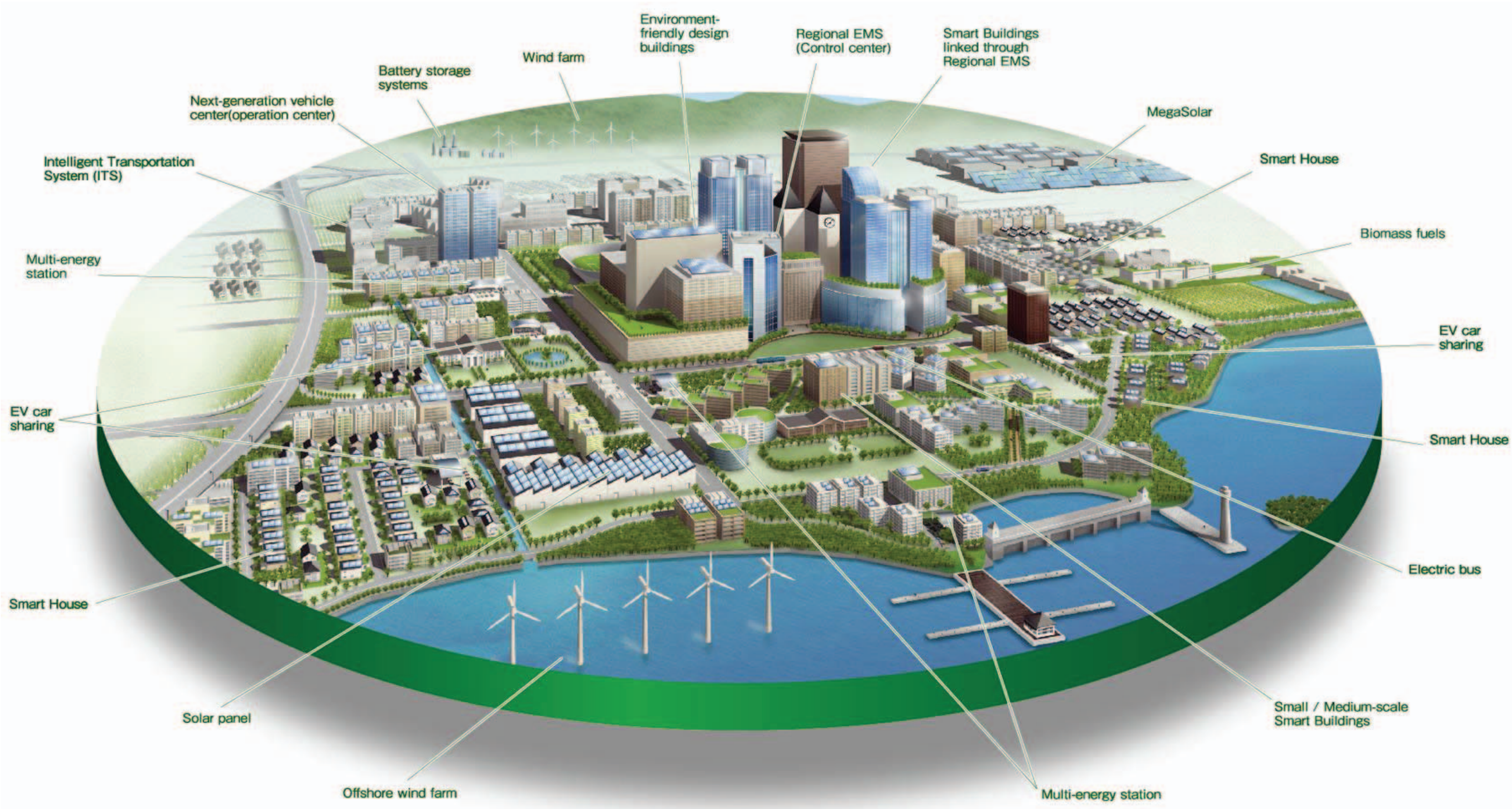
INTRODUCTION

What is a Smart City?

- Definition - 1: A city "connecting the physical infrastructure, the information-technology infrastructure, the social infrastructure, and the business infrastructure to leverage the collective intelligence of the city".
- Definition - 2: "A smart sustainable city is an innovative city that uses information and communication technologies (ICTs) and other means to improve quality of life, efficiency of urban operations and services, and competitiveness, while ensuring that it meets the needs of present and future generations with respect to economic, social and environmental aspects".

Source: Mohanty 2016, CE Magazine July 2016

Broad Picture of Smart City



Source: http://edwingarcia.info/2014/04/26/principal/

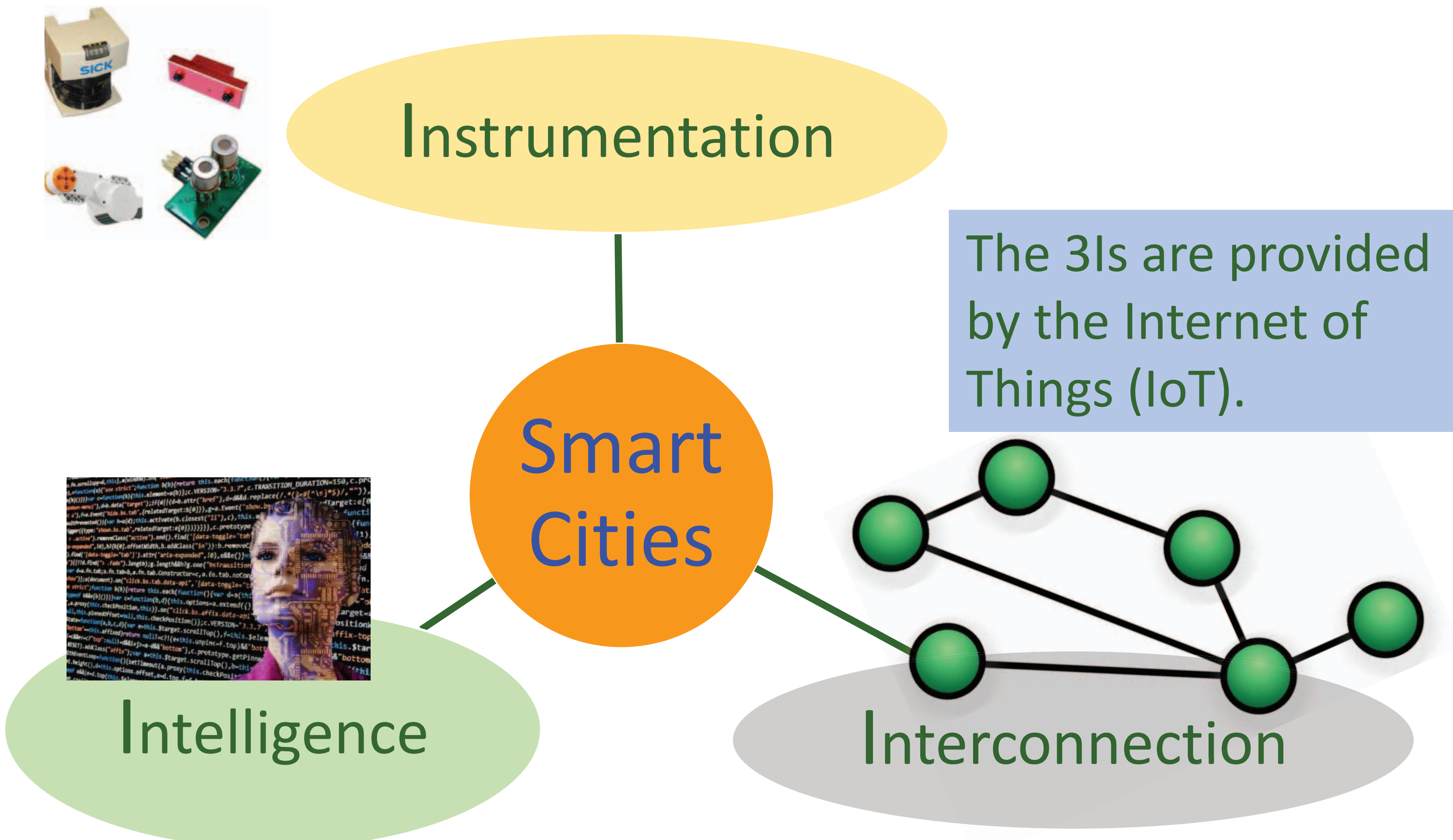
Idea of Smart City

Smart Cities ←

Regular Cities

+ Information and Communication Technology (ICT)
+ Smart Components
+ Smart Technologies

Components of Smart City



Source: Mohanty 2016, EuroSimE 2016 Keynote Presentation

Technology in Smart City

Things
Sensors/actuators with IP address that can be connected to Internet

Local Network
Can be wired or wireless: LAN, Body Area Network (BAN), Personal Area Network (PAN), Controller Area Network (CAN)

Cloud Services
Data either sent to or received from cloud (e.g. machine activation, workflow, and analytics)

Global Network
Connecting bridge between the local network, cloud services and connected consumer devices

Connected Consumer Electronics
Smart phones, devices, cars, wearables which are connected to the Things

Overall architecture:
- A configurable dynamic global network of networks
- Systems-of-Systems

Source: Mohanty ICIT 2017 Keynote

Smartness of a Smart City

- Smartness in a Smart-City is not just installing digital interfaces in infrastructure but also using technology and data provided to enhance and improve the quality of life.
- Smart-City is a system of three layers: infrastructure, smart-devices or applications which are capable of data analysis and the technology which includes connected networks of devices and sensors.
- Technological innovations could possibly provide a solid foundation to smart cities.

MOTIVATION

How much is too Smart?

- The smartness of a smart city is still a rough area as there is not one specific answer for how much is too smart.
- To be able to improve the quality of life of users and to enable them with an option of taking control over their lives should be also considered an important factor for any city to be smart.

Motive of iMirror

- For any organization to be successful, key factors are the employees and managers working for it. If the manpower is not mentally strong, then progress may not be up to the mark.
- In the current lifestyle, employees, students, teachers, and employers are most affected by the pressures they experience, which leads to stress.
- For a strong foundation of smart-city, having mentally healthy employees is important, as they drive the progress.
- In order to address the work stress among individual before or after pandemics like COVID-19, this research proposes iMirror which analyzes the stress levels of a person and gives feedback to users in order to maintain a healthy stress response system.

Commercial Use of iMirror

❖ A happy businessman keeping stress levels in control with the use of iMirror for a better growth, both personally and professionally.

IMPORTANCE OF STRESS

Importance of Stress

Distress

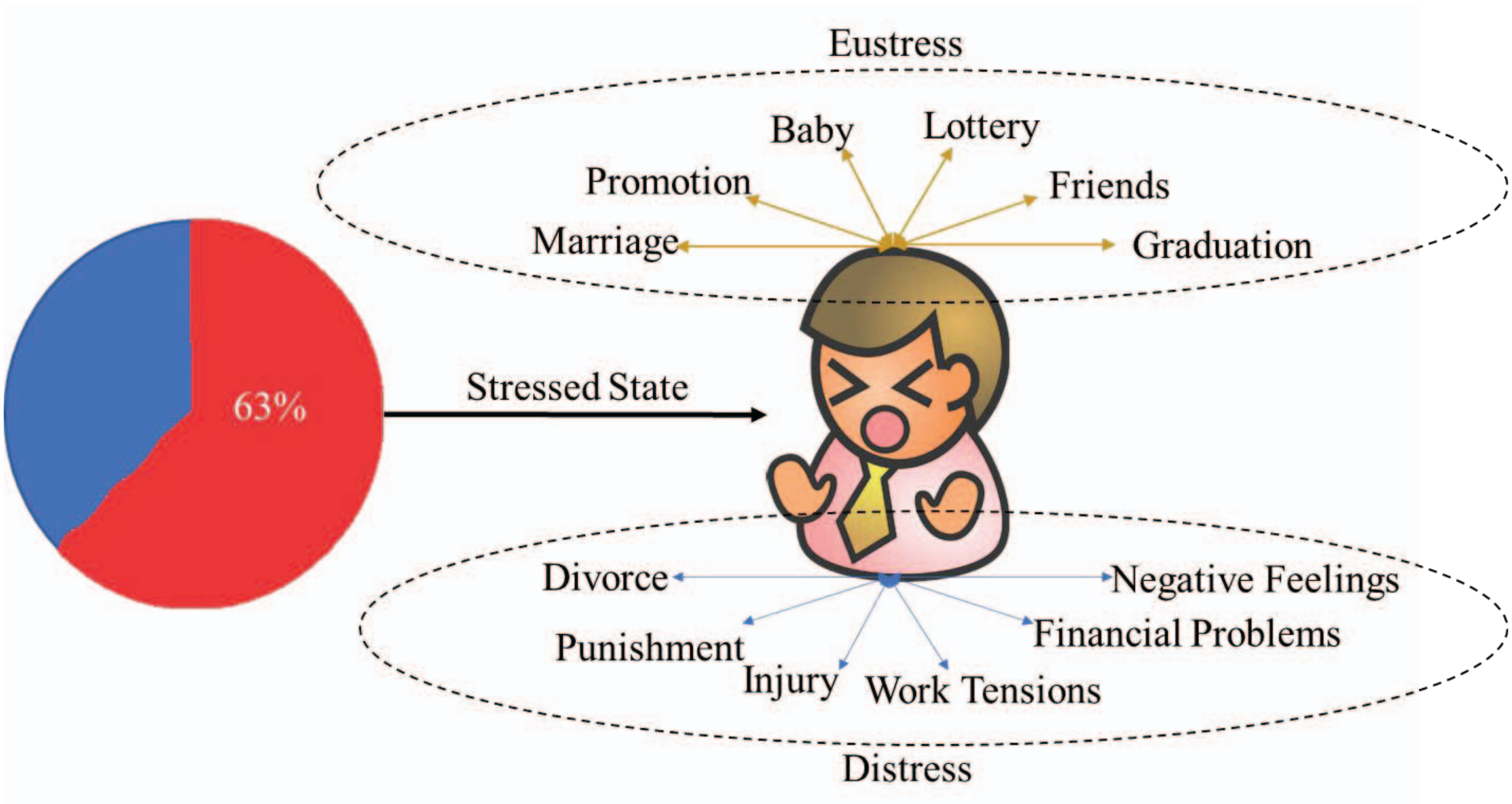
When there is an encounter with sudden **stress**, the brain floods the **body** with chemicals and hormones such as adrenaline and cortisol.

Eustress

- Lack of Energy
- Type 2 Diabetes
- Osteoporosis
- Mental cloudiness (brain fog) and memory problems
- A weakened immune system

Stress is the body's reaction to any change that requires an adjustment or response.

Stressors of Stress



Symptoms of Stress

EXISTING SOLUTIONS – THEIR PROBLEMS

STATE-OF-THE-ART LITERATURE

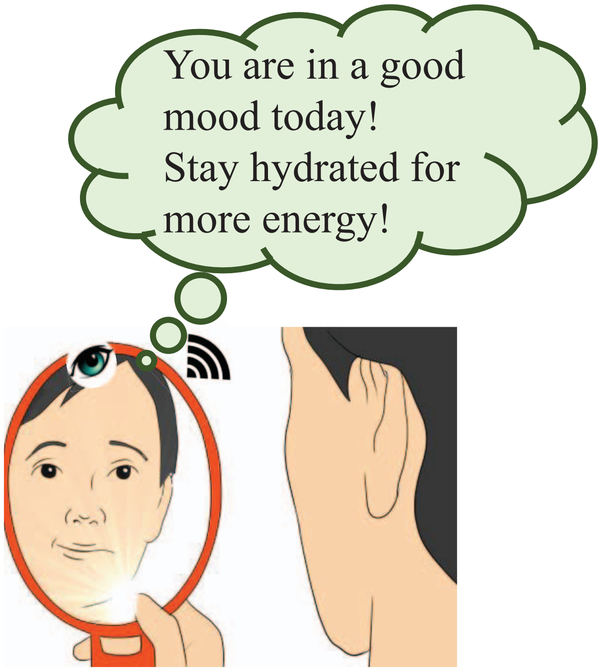
Research	Input	Action	Drawbacks
Giannakakis et al.	Video	Emotion Analysis	No User Interface, No user access, No real time processing
Liao et al.	Physiological Sensor Data	Stress Monitoring	User input required, No User Interface, No user access, No real time processing
Gong et al.	Images	Emotion Analysis	No Real time monitoring, no relationship with stress is established, no data visualization for user
Huang et al.	Images and EEG data	Emotion Analysis	Stress monitoring is not performed, No User interface, No user access, No real time processing
Luoh et al.	Images	Emotion Analysis	No User interface, No user access, No real time processing, No stress detection
iMirror (Current Paper)	Images	Stress Analysis	User Interface, User access, Real time processing, Stress Detection

Research Question:

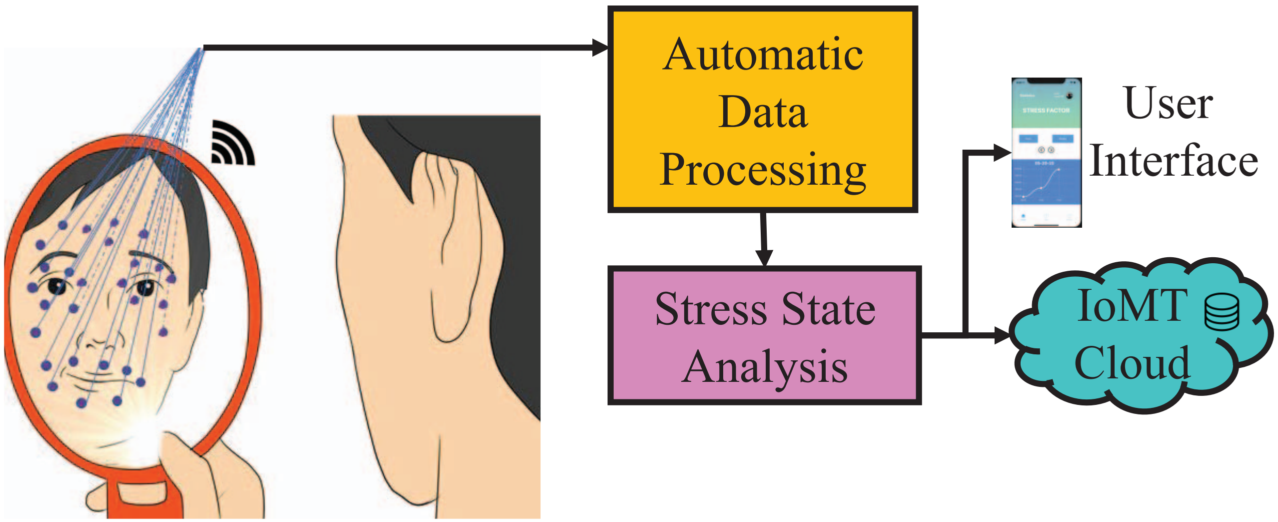
- How to have an accurate and rapid Stress Level Detection system that *analyzes facial features, and detects stress level* at the user end (at *IoT-Edge*) and stores the data at the cloud end (at *IoT-Cloud*)?

PROPOSED SOLUTION

Device Prototype of iMirror



Broad Perspective of iMirror



Broad Perspective of iMirror

- Whenever there is a face detected, the iMirror system starts the process of obtaining the facial features of the user.
- Once these images with the features are obtained, they are sent for automatic feature processing.
- The extracted information is sent to the stress state analysis unit.
- This again is connected to the user interface for the stress control remedies.

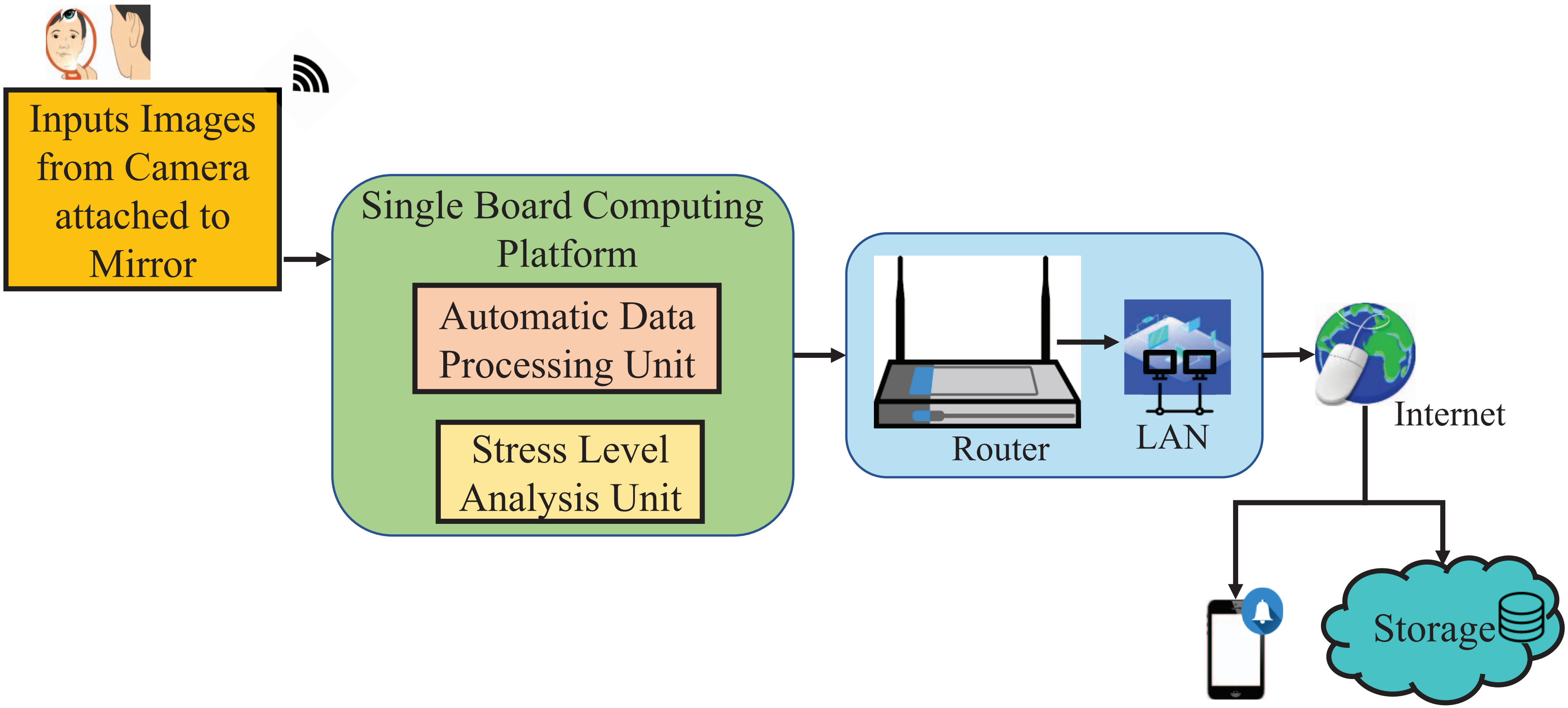
NOVEL CONTRIBUTIONS

- A real time stress monitoring system with no user input.
- An interface which uses real time location of the user to provide appropriate remedies.
- A potential extension of any product that gathers the data from the user.
- A potential marketable product that can be placed anywhere for monitoring purposes.
- A methodology that can address immediate control of stress.
- Visualization tools for the user for easy data comprehension.
- Allows users to understand the fluctuations of stress and get back control over their body.

ARCHITECTURE OF iMIRROR

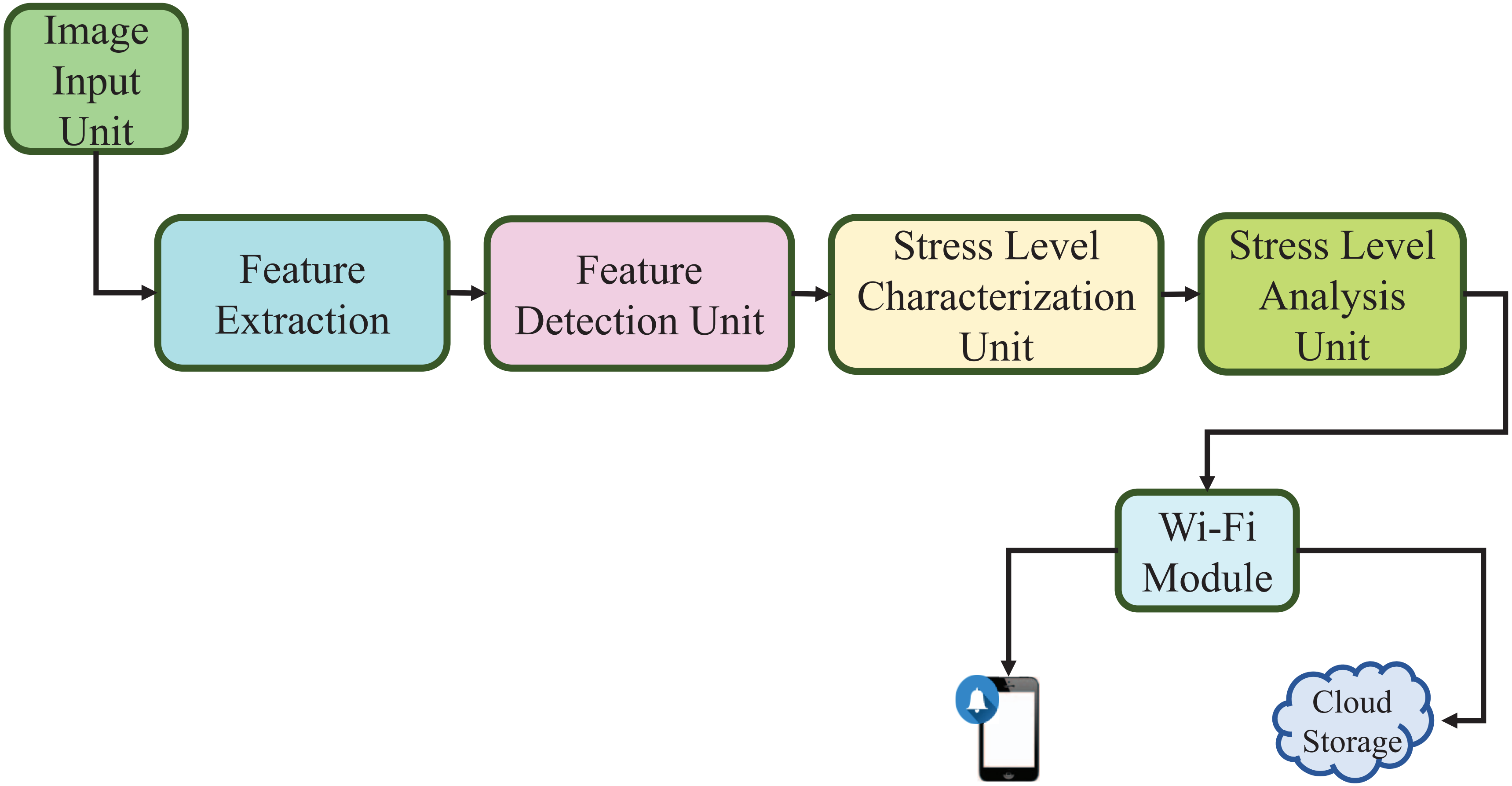
- The input image data is taken from the camera and sent to an SBC which could be located on-site or located remotely and be a part of the network.
- The object classification is performed using graphical annotation tools followed by feature extraction by segmenting the background data.
- From there, the objects or features are detected and are then considered for the stress state analysis of the user.
- The SBC relates to a cloud platform which hosts the database.
- The database is connected to the mobile application which is used as a user interface in iMirror.

Overall Architecture



PROPOSED METHODOLOGY

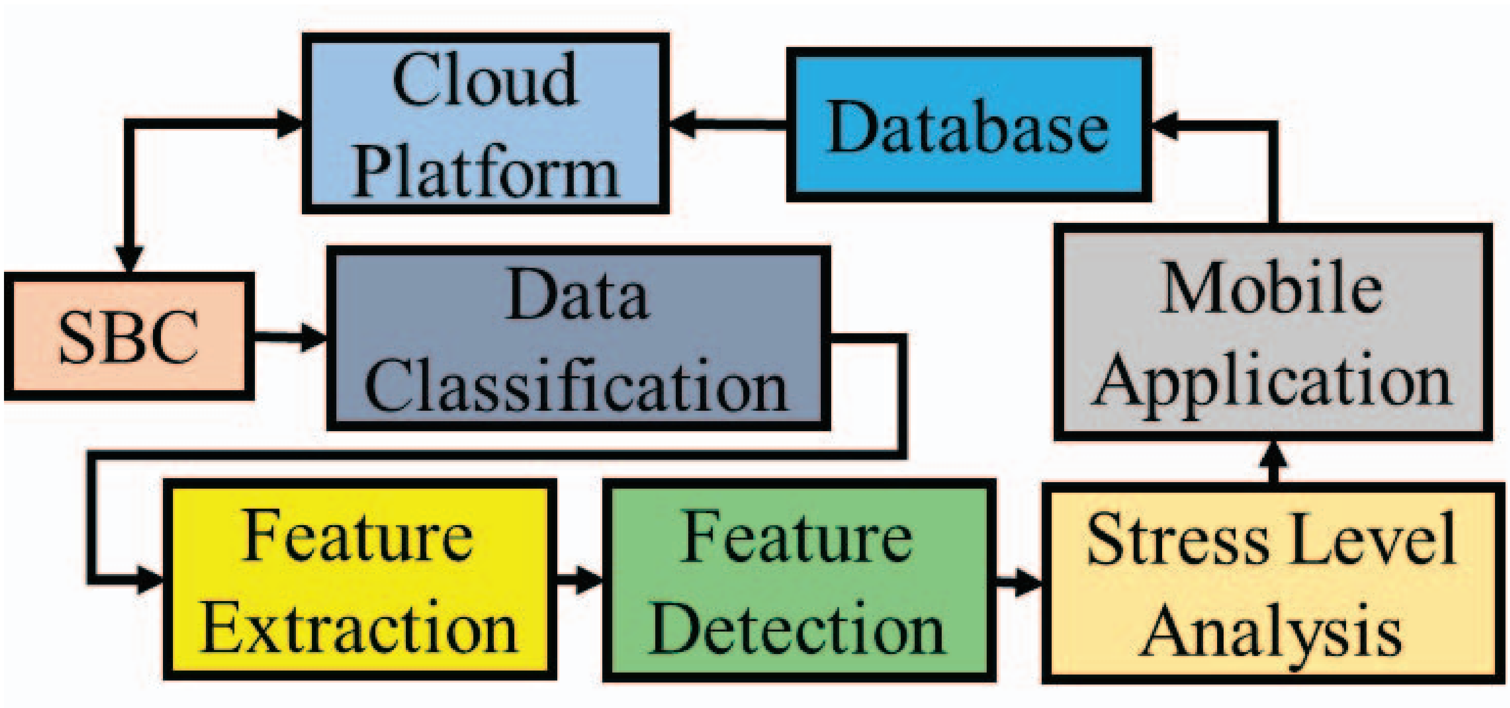
Automatic Data Processing



- The images from the wearable proposed are taken at the input stage.
- Once the images are obtained, they are processed for feature extraction and detection of the features that are focused in iMirror.
- After the features are obtained, along with their confidence levels, the stress is characterized.
- This characterized stress is used for the stress analysis.
- This analyzed data is sent to the user for feedback and stress control remedies.

SYSTEM LEVEL MODELING

❖ System Level Modeling for Stress Detection in iMirror.



Feature Classification & Detection

1. Images which are to be used for testing and training the model are collected.
2. The formats of the images are converted from JPEG to XML after creating bounding boxes by using any graphical image annotation tool.
3. Multiple bounding boxes in various images for the same feature, which are called priors, are also created in the same annotation tool.
4. Using box-coder, the dimensions of priors are made equal.
5. By considering the concept of IOU (Intersection Over Union), the matched and unmatched thresholds for matching the ground truth boxes to priors are set.
6. Images in XML format are made equal in size either by using reshape or resize functions.
7. Using convolution and rectified linear functions, the feature maps are assigned to every image sent to the model.
8. Based on these features, the images are either sent to regression or classification where the objects are detected through boxes in the images.
9. Repeat the above steps for all the images.

Stress Level Characterization

1. When there is a person observed in front of the mirror by motion sensor m, declare and initialize a input variable id and timer t to zero.
2. Initialize the confidence values of feature variables r, e, d, f, s to zero.
3. Initialize the output variable sl to zero.
4. **while** m ! = 0 **do**
5. Update t and id.
6. Update r, e, d, f, s.
7. **end while**
8. **if** 80% < = r, e, d, f, s < = 100% **then**
9. Update sl to High.
10. **else if** 60% < = r, e, d, f, s < = 70% **then**
11. Update sl to Medium.
12. **else if** 50% < = r, e, d, f, s < = 60% **then**
13. Update sl to Low.
14. **else**
15. Update sl to Normal.
16. **end if**
17. **while** m ! = 0 ’and’ id == id **do**
18. Update id1 ++ and t1 ++.
19. Repeat from Steps 2 through 16.
20. **end while**

Stress Level Analysis

1. Declare and initialize final stress fs to zero.
2. Take the updated sl, t and id.
3. Declare and initialize the variable nt for number of iterations of t.
4. **if** id ! = 0 **then**
5.   **if** t == 0 **then**
6.   Update fs.
7.   **else**
8.   Compute Sum for sl at all t and store it in fs.
9.   Divide fs with nt.
10.   Update fs.
11.   **end if**
12. **end if**

Metrics in iMirror

- Precision: The ability of a model to identify only the relevant objects from an image is known as its precision (P):

$$P = \left(\frac{TP}{TP + FP} \times 100\% \right)$$

- Recall: The ability of a model to identify all the relevant cases from the detected relevant objects is called recall (R):

$$R = \left(\frac{TP}{TP + FN} \times 100\% \right)$$

Metrics in iMirror

- Confidence: Confidence is used to rank the predictions and quantify the relationship between prediction and recall as we consider increasing numbers of lower ranked predictions. Instead of presenting a single error code, a confidence interval CI is calculated:

$$CI = z\sqrt{\left(\frac{\alpha \cdot (1-\alpha)}{N_{sample}}\right)},$$

where N_{sample} is the sample size, z is a critical value from the Gaussian distribution, and α is the accuracy obtained.

Metrics in iMirror

- Accuracy: The accuracy of a model can be defined as the ratio of correct detection made by the model to all the detection's made by the model:

$$\alpha = \left(\frac{TP + TN}{TP + TN + FN + FP} \right) \times 100\%$$

- True Positive (TP): A correct detection.
- False Positive (FP): A wrong detection.
- False Negative (FN): This occurs when a ground truth is not detected.
- True Negative (TN): This is the possible outcome where the model correctly predicts a ground false.

iMIRROR IMPLEMENTATION

Machine Learning Model

- The model has been trained with 1000 images. Out of these 1000, 800 have been used for training the model and the remaining 200 have used to test the model's confidence rates.
- The model that has been used here is SSD (Single Shot MultiBox Detector) Mobilenet as this is the only one that has the capability of working in a lightweight version.
- The TensorFlow object Detection API has been used. The prompt with 33289 steps and 9704 learning is represented.
- There are 7 layers with 32 batch sizes for every iteration. The initial learning rate assigned is 0.001. Rectified Linear Function is used as the activation function in all layers.

Implementation

- Training the Model
  - Initial steps of the process.

```
Anaconda Prompt (Anaconda3) - python  train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/ssd_mobilenet_...
I0310 21:32:12.388569  2968 learning.py:507] global step 191: loss = 5.9407 (1.348 sec/step)
INFO:tensorflow:global step 192: loss = 5.9650 (1.346 sec/step)
I0310 21:32:13.739798  2968 learning.py:507] global step 192: loss = 5.9650 (1.346 sec/step)
INFO:tensorflow:global step 193: loss = 6.2748 (1.333 sec/step)
I0310 21:32:15.076202  2968 learning.py:507] global step 193: loss = 6.2748 (1.333 sec/step)
INFO:tensorflow:global step 194: loss = 6.2461 (1.321 sec/step)
I0310 21:32:16.403846  2968 learning.py:507] global step 194: loss = 6.2461 (1.321 sec/step)
INFO:tensorflow:global step 195: loss = 6.4576 (1.341 sec/step)
I0310 21:32:17.748458  2968 learning.py:507] global step 195: loss = 6.4576 (1.341 sec/step)
INFO:tensorflow:global step 196: loss = 6.3665 (1.358 sec/step)
I0310 21:32:19.109527  2968 learning.py:507] global step 196: loss = 6.3665 (1.358 sec/step)
INFO:tensorflow:global step 197: loss = 6.0075 (1.330 sec/step)
I0310 21:32:20.442073  2968 learning.py:507] global step 197: loss = 6.0075 (1.330 sec/step)
INFO:tensorflow:global step 198: loss = 6.1272 (1.335 sec/step)
I0310 21:32:21.781166  2968 learning.py:507] global step 198: loss = 6.1272 (1.335 sec/step)
INFO:tensorflow:global step 199: loss = 5.6404 (1.314 sec/step)
I0310 21:32:23.099686  2968 learning.py:507] global step 199: loss = 5.6404 (1.314 sec/step)
INFO:tensorflow:global step 200: loss = 6.2505 (1.328 sec/step)
I0310 21:32:24.431737  2968 learning.py:507] global step 200: loss = 6.2505 (1.328 sec/step)
INFO:tensorflow:global step 201: loss = 6.4575 (1.393 sec/step)
I0310 21:32:25.836921  2968 learning.py:507] global step 201: loss = 6.4575 (1.393 sec/step)
INFO:tensorflow:global step 202: loss = 6.4877 (1.499 sec/step)
I0310 21:32:27.345548  2968 learning.py:507] global step 202: loss = 6.4877 (1.499 sec/step)
INFO:tensorflow:global step 203: loss = 5.3990 (1.432 sec/step)
I0310 21:32:28.780710  2968 learning.py:507] global step 203: loss = 5.3990 (1.432 sec/step)
INFO:tensorflow:global step 204: loss = 5.0332 (1.370 sec/step)
I0310 21:32:30.156709  2968 learning.py:507] global step 204: loss = 5.0332 (1.370 sec/step)
INFO:tensorflow:global step 205: loss = 6.0380 (1.371 sec/step)
I0310 21:32:31.533221  2968 learning.py:507] global step 205: loss = 6.0380 (1.371 sec/step)
```

Implementation

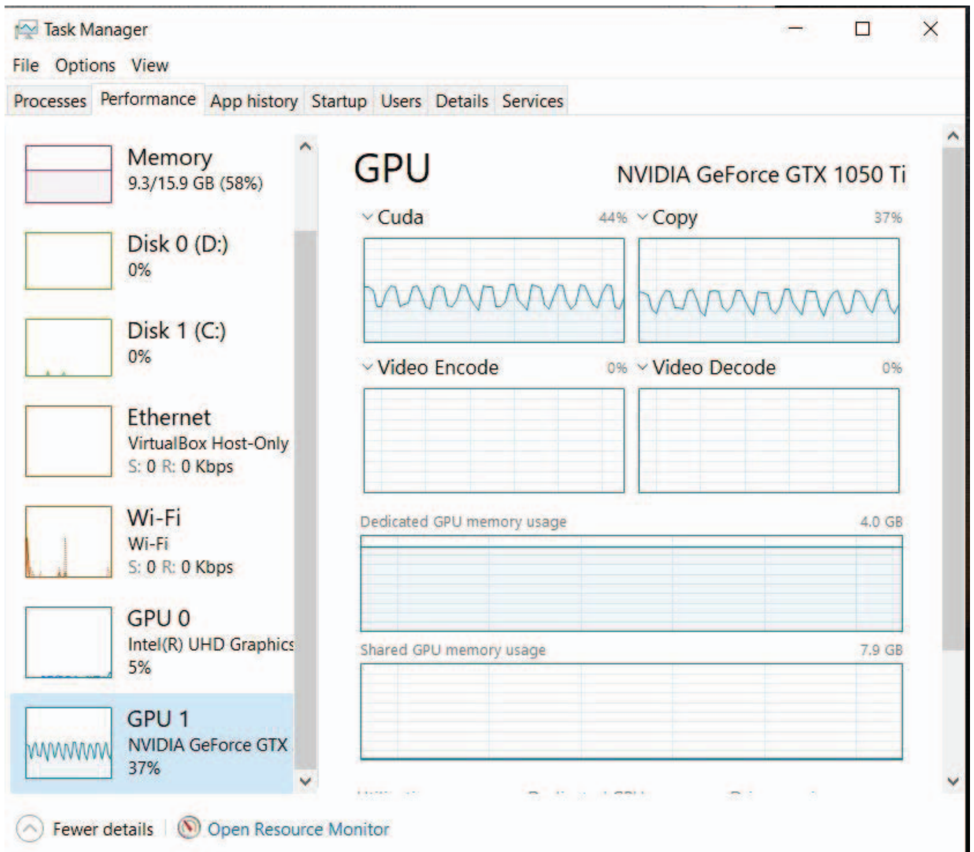
- The model has been trained up to 33289 steps with a loss <1%.

Anaconda Prompt (Anaconda3)

```
I0311 20:23:00.850183  9704 learning.py:507] global step 33275: loss = 1.0213 (1.171 sec/step)
INFO:tensorflow:global step 33276: loss = 0.7677 (1.161 sec/step)
I0311 20:23:02.012160  9704 learning.py:507] global step 33276: loss = 0.7677 (1.161 sec/step)
INFO:tensorflow:global step 33277: loss = 0.8371 (1.160 sec/step)
I0311 20:23:03.174136  9704 learning.py:507] global step 33277: loss = 0.8371 (1.160 sec/step)
INFO:tensorflow:global step 33278: loss = 1.0600 (1.165 sec/step)
I0311 20:23:04.342130  9704 learning.py:507] global step 33278: loss = 1.0600 (1.165 sec/step)
INFO:tensorflow:global step 33279: loss = 0.8597 (1.155 sec/step)
I0311 20:23:05.499120  9704 learning.py:507] global step 33279: loss = 0.8597 (1.155 sec/step)
INFO:tensorflow:global step 33280: loss = 1.0978 (1.163 sec/step)
I0311 20:23:06.665052  9704 learning.py:507] global step 33280: loss = 1.0978 (1.163 sec/step)
INFO:tensorflow:global step 33281: loss = 1.3743 (1.176 sec/step)
I0311 20:23:07.841988  9704 learning.py:507] global step 33281: loss = 1.3743 (1.176 sec/step)
INFO:tensorflow:global step 33282: loss = 0.7632 (1.184 sec/step)
I0311 20:23:09.029931  9704 learning.py:507] global step 33282: loss = 0.7632 (1.184 sec/step)
INFO:tensorflow:global step 33283: loss = 0.8571 (1.174 sec/step)
I0311 20:23:10.206836  9704 learning.py:507] global step 33283: loss = 0.8571 (1.174 sec/step)
INFO:tensorflow:global step 33284: loss = 1.1365 (1.190 sec/step)
I0311 20:23:11.398733  9704 learning.py:507] global step 33284: loss = 1.1365 (1.190 sec/step)
INFO:tensorflow:global step 33285: loss = 1.3379 (1.387 sec/step)
I0311 20:23:12.788981  9704 learning.py:507] global step 33285: loss = 1.3379 (1.387 sec/step)
INFO:tensorflow:global step 33286: loss = 0.9948 (1.174 sec/step)
I0311 20:23:13.964921  9704 learning.py:507] global step 33286: loss = 0.9948 (1.174 sec/step)
INFO:tensorflow:global step 33287: loss = 0.7842 (1.152 sec/step)
I0311 20:23:15.118918  9704 learning.py:507] global step 33287: loss = 0.7842 (1.152 sec/step)
INFO:tensorflow:global step 33288: loss = 1.1423 (1.142 sec/step)
I0311 20:23:16.264301  9704 learning.py:507] global step 33288: loss = 1.1423 (1.142 sec/step)
INFO:tensorflow:global step 33289: loss = 0.9563 (1.163 sec/step)
I0311 20:23:17.430224  9704 learning.py:507] global step 33289: loss = 0.9563 (1.163 sec/step)
Traceback (most recent call last):
```

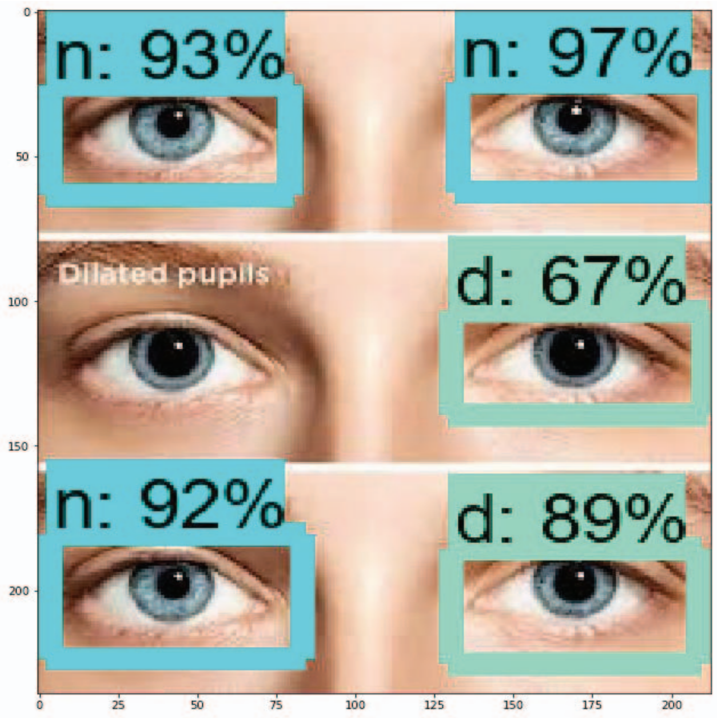
Implementation

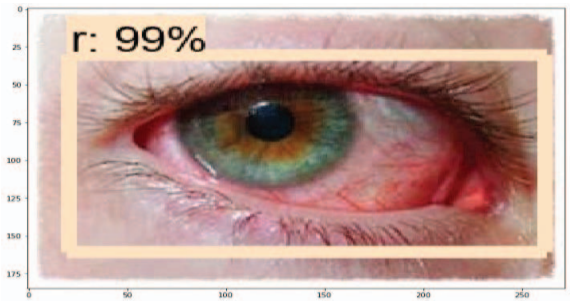
- PC Task Manager View

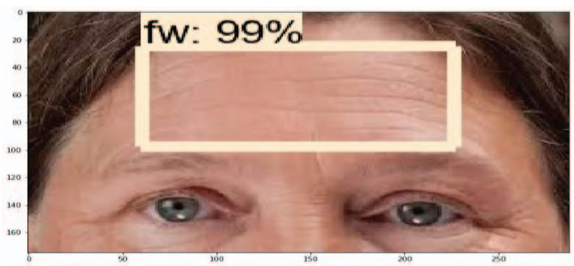


Results of iMirror

- TensorFlow Implementation:
  The confidence rate for various classes of features considered is displayed along with the bounding box in the image.







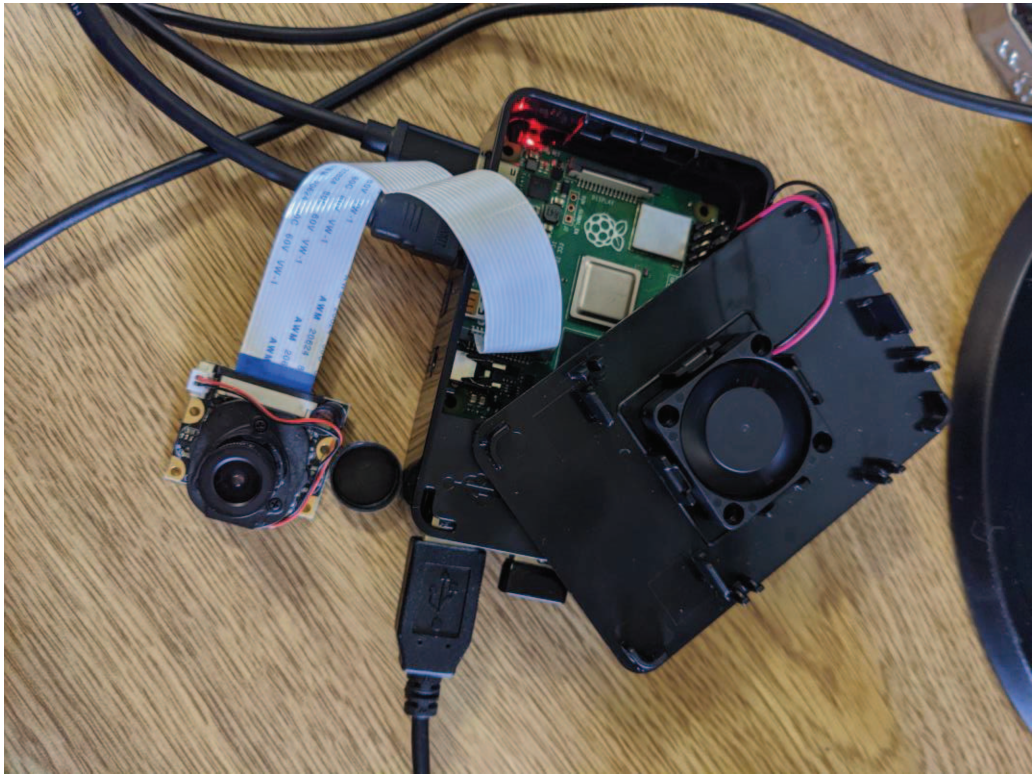
iMIRROR IMPLEMENTATION ON SBC

Lite Machine Learning Model

- TensorFlow Lite is a reduced lightweight version of TensorFlow which is focused on mobile and embedded devices applications.
- The iMirror model has been implemented with TF-Lite as this produced a higher frame per second (FPS) rate.
- For training and testing, 350 images were used, out of which 280 images were used for training while 70 are used for testing.

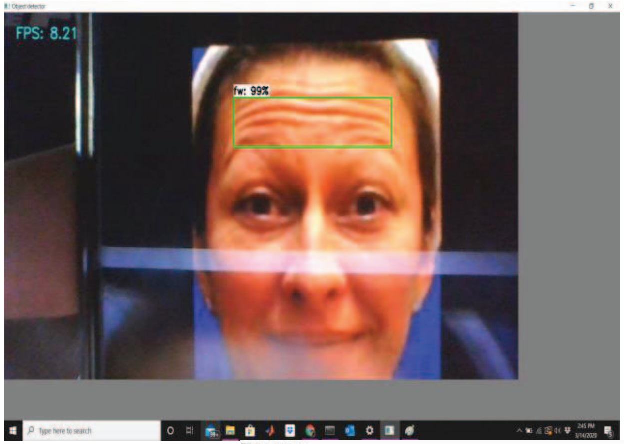
Hardware Implementation

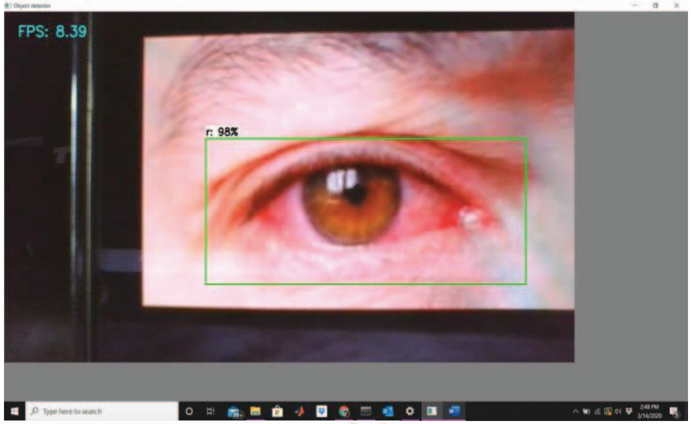
- Raspberry Pi 4 Model B+

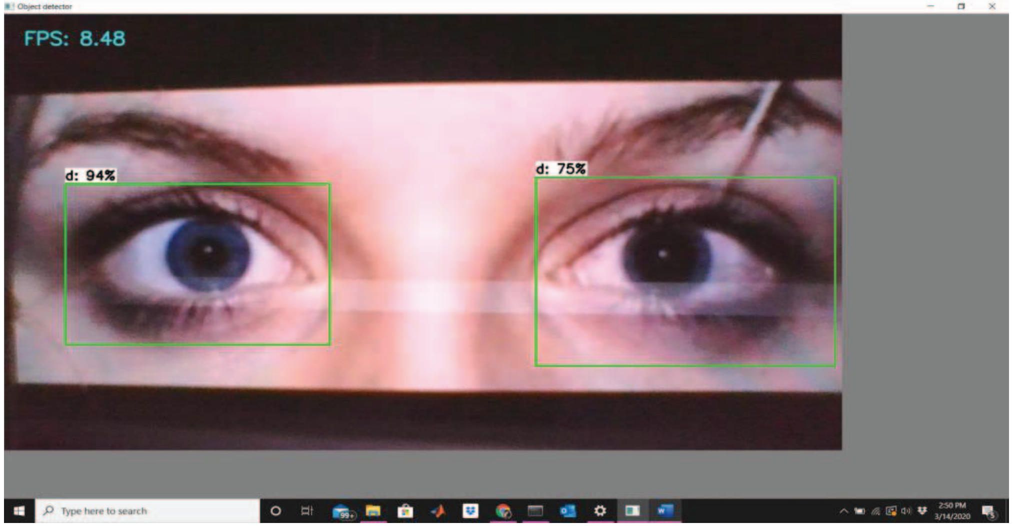


Results of iMirror on SBC

- TensorFlow Lite Implementation







ACCURACY RESULTS

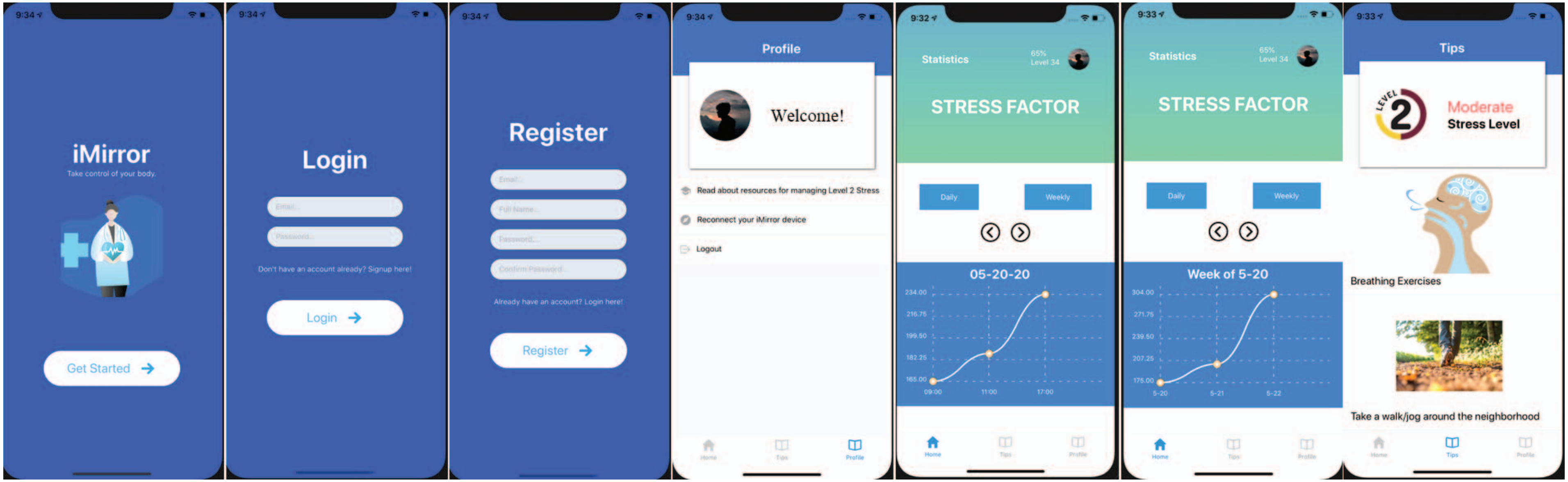
Feature	CI(%) for 5-fold cross validation with 15 epochs	CI for 5 Repeated 5-fold cross validation with 15 epochs
Facial Sweat	88	96
Eye Redness	89	97
Forehead Frown	88	94
Pupil Dilation	89	97
Eye Bags	85	93

iMIRROR IMPLEMENTATION FROM SBC TO USER INTERFACE

- After the stress detection and analysis is performed, for the user to access the information, the data from the SBC is sent to the mobile application.
- The SBC is connected to the cloud platform which is again connected to the database.
- This database is connected to the mobile application simulator where the application was built.
- The tools which were used in developing this mobile application are React Native, Firebase Cloud Services, Tailwind CSS, Expo, React-native-chart-kit along with Visual Studio Code and iOS Simulator.

Mobile Application as an UI

❖ The mobile application was built on the mobile application simulator.



CHARACTERISTICS OF iMIRROR

❖ The characteristics that are used in the implementation of iMirror are summarized.

Characteristics	Specifics
Input System	Images (JPEG) from Camera
Data Acquisition	Database and API approach
Data Analysis Tool	TensorFLow and TensorFlow Lite
Graphical Annotation Tool	LabelImg
Input Dataset	1000 and 350 images
Classifier	SSD MobileNET
Features Considered	6
Accuracy	97% and 98% on single board platform

SPECIFICATIONS OF iMIRROR

- The specifications of the model along with its implementation type are presented

Characteristics	Single Board Computer
Camera	5MP; 640x480
Accuracy	98%
Average Precision	81.2%
Object Detection	Yes
Object Classification	Yes
Stress Level Characterization	Yes
Input Type	Images
Automation	Fully Automated

CONCLUSION

COMPARATIVE ANALYSIS

Research	Input	ML Algorithm	Features Extracted	Stress Levels	Accuracy (%)
Kolodziej, et al. [24]	Images	KNN, SVM, BT	6	No	52.8, 55.9, 57.7
Tarnowski, et al. [25]	Images	KNN, MLP	7	No	73
Du, et al. [26]	Images	KSDA	7	No	83.2
Luoh, et al. [18]	Images	GMM	N/A	No	90
Huang, et al. [17]	Images	KNN	N/A	No	N/A
iMirror (Current Paper)	Images	SSD Mobilenet	5	4	97

Conclusions

- The approach presented here provides an extension to monitoring systems by focusing on facial features of the users and analyzing if the person is stressed or not.
- The accuracy of detection is found to be 97%, which strongly suggests this approach is suitable for effectively monitoring the stress state of a person.
- The approach could be an answer to a long-time sought-after need for watching the emotional behaviors and their impact on overall physical and mental health.

FUTURE RESEARCH

- The broad adaptation of iMirror is what we aim through this research, as iMirror could be an important component among employers and employees thereby allowing the mankind to have healthy stress response systems.
- By allowing people to have stable mental health, iMirror strives to achieve the true state of “Smartness” for any Smart-City.
- Introduce security in order to provide a secure backbone for the smart healthcare industries.
- Incorporate more stressors for stress detection such as facial expression changes, posts in social media, fluctuations in emotional state of a person etc.,.

Questions?

Thank you!