DE-40(EE)

NUST COLLEGE OF

ELECTRICAL AND MECHANICAL ENGINEERING

SMART BABY MONITORING SYSTEM

A PROJECT REPORT

DE-40 (EE)

Submitted by

NC Hamza Khalil
NC Hassan Ali
NC M. Hassan Touqir

BACHELORS IN ELECTRICAL ENGINEERING
YEAR 2022

PROJECT SUPERVISOR

Dr. Ahmad Rauf Subhani

NUST COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING
PESHAWAR ROAD, RAWALPINDI

CERTIFICATE OF APPROVAL

It is to certify that the project " Smart Baby Monitoring System" was done by NC Hamza Khalil, NC Hassan Ali and NC M. Hassan Touqir , under the supervision of Dr. Ahmad Rauf Subhani.

This project is submitted to Department of Electrical Engineering, College of Electrical and Mechanical Engineering (Peshawar Road Rawalpindi), National University of Sciences and Technology, Pakistan in partial fulfillment of requirements for the degree of Bachelors of Engineering in Electrical Engineering.

Students:

1- **Hamza Khalil**

NUSTID: _____ Signature: _____

2- **Hassan Ali**

NUSTID: _____ Signature: _____

3- **M. Hassan Touqir**

NUSTID: _____ Signature: _____

APPROVED BY:

Project Supervisor: _____ Date: _____

Dr. Ahmad Ruaf Subhani

DECLARATION

We hereby declare that no portion of the work referred to in this Project Thesis has been submitted in support of an application for another degree or qualification of this of any other university or other institute of learning. If any act of plagiarism found, we are fully responsible for every disciplinary action taken against us depending upon the seriousness of the proven offence, even the cancellation of our degree.

1- Hamza Khalil

NUSTID: _____ Signature: _____

2- Hassan Ali

NUSTID: _____ Signature: _____

3- M. Hassan Touqir

NUSTID: _____ Signature: _____

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

First and foremost, we are thankful to Allah Almighty, the most merciful and the most beneficent, for the blessings, courage, and the intellect He has bestowed us with, because of which it became possible for us to accomplish such remarkable goal.

We would like to express our deepest gratitude to our supervisor, Dr. Ahmad Rauf Subhani for his dedicated guidance, invaluable advice, and constant encouragement throughout our bachelor study. We are also indebted to him for the efforts he has devoted to serious consultations of this project. His enthusiasm and insights in many research problems have provided me with a source of thoughts and actions.

We are thankful to our parents and the family for their constant support and encouragement at each thick and thin. Moreover, we are grateful to all our fellows, friends who supported us intellectually and emotionally. Many thanks for all those who contributed in it, without all their support and appreciation we would be lacking a major source of inspiration and motivation. We are hopeful that this project will add value to the people's life and would be developed further to adopt this technology on the massive level acting as a benchmark for future technologies of Pakistan.

ABSTRACT

The aim of the project is to facilitate new parents with their new babies. As they have little to no experience in taking care of their babies. Moreover, working parents would hugely benefit from our proposed solution.

The project comprises of 2 parts, firstly cry of the baby is classified into four classes namely, Hunger, Tiredness, Discomfort and Belly-Pain. Machine Learning techniques are used for cry classification.

Second part of the project is regarding baby monitoring.
Pi cam and Raspberry Pi 4 are used to monitor the baby such as if the baby is sleeping or awake. Pose estimation and Eye Aspect Ratio are used for Sleep/Awake Detection. If eyes of the baby are visible then EAR ratio is used and if eyes are not visible then Pose Estimation is used.

TABLE OF CONTENTS

LIST OF FIGURES

CH 1: INTRODUCTION

1.1. Background

About 130 million babies are born globally each year. Taking good care of newborns is a big challenge, especially for first time parents. Following the suggestions from other parents and books is not enough to solve the problems in practice. The main reason is because it is difficult to understand the meaning of the infant cries. Infants communicate with the world through crying. Experienced parents, caregivers, doctors, and nurses understand the cries based on their experiences. Young parents get frustrated and have trouble calming down their babies because all cry signals sound the same to them. Accurately interpreting infants' cry sound can help parents take better care of their babies. Research on infant cry started as early as 1960s when Wasz –Hockert research group identified the four types of the cries (pain, hunger, birth, and pleasure) auditorily by trained nurses.

In the early years, researches have determined that different types of cries can be differentiated auditorily by trained adult listeners. But training human perception for infant cry is much harder than training machine learning models. In Mukhopadhyay's study, the highest classification accuracy by training a group of people to recognize some cry sounds is 33.09% while machine learning algorithm based on spectral and prosodic features can recognize the same set of data and reach 80.56% accuracy. Building smart machines to understand infant cry leads the way to build intelligent robot caregivers in the future. Besides understanding infants' daily life needs, disease prediction is another critical task in infant cry research. Since infants' vocal tract and breathing system are affected by some diseases, the cry signals of unhealthy infants contain unique characteristics that differ from healthy cry signals. Known examples of such diseases include deaf, autism, and asphyxia, etc. Analyzing pathological cry signals to identify diseases is a non-invasive and fast method that can save infants' lives, especially in the areas that lack of medical equipment and expertise. In the early years of infant cry research, many works have focused on classifying normal and pathological cry signals. In Saraswathy's review, 34 papers on classification of normal and

pathological cry signals published from 2003 to 2011 are listed. The works include identifying diseases such as hypo-acoustic, asphyxia, hypothyroidism, hyperbilirubinemia, cleft palate, etc.

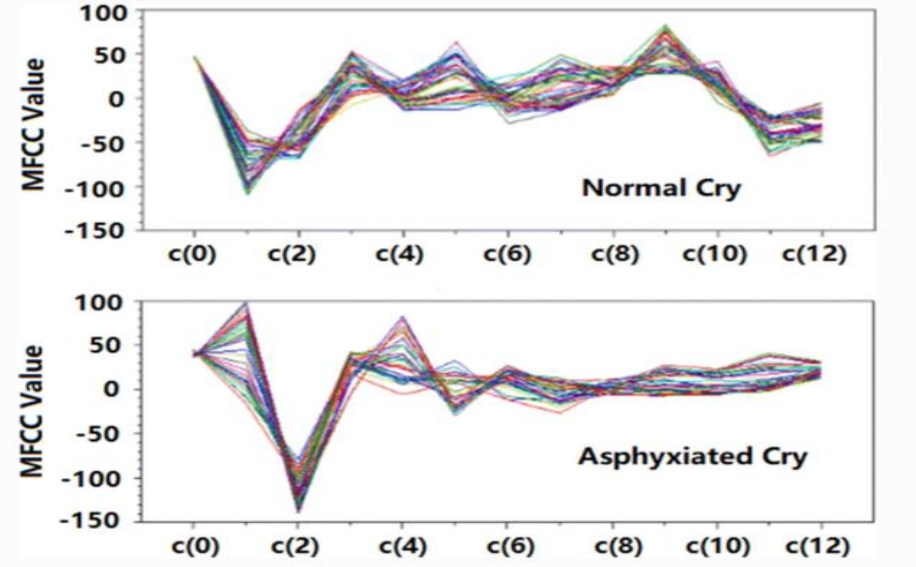
Infant cry research involves data collection, cry signal processing, feature extraction and selection, and classification. Due to the sensitivity of cry data, it has been difficult for researchers to acquire data needed. Researchers either record cry clips by themselves or ask permissions for datasets from other authors. Most databases are recorded in hospital, Neonatal Intensive Care Unit (NICU), home, and clinics, etc. by recording in real time or by setting up electronic recording devices close to the infants' crib for long period of time. Signal processing is a must to remove background noises and perform cry segmentation to build cry databases. Once the database is available, feature extraction is the step to extract features from different domains of the cry signals. Features extracted from time domain, cepstral domain, or prosodic domain, etc. represent different aspects of the cry signal. Selecting the most appropriate features and reducing the feature dimensions are another task to build effective classification models. Applying appropriate machine learning models for specific cry features is vital for classification or detection accuracy. As the second Artificial Intelligence (AI) winter ends in 1990s, neural networks emerge as a popular method in infant cry research. Neural networks are computing system, containing interconnected neurons, inspired by biological brain system. Input vectors, neurons, weights, activation functions, and output are the main elements in a neural network. Each neuron has a value computed in the forward propagation process based on the weights of each connection and bias of each layer. Activation functions are used to achieve nonlinearity in the network. The back propagation is the key algorithm to train the model and minimize the loss function, which evaluates how well the model fits the dataset. During the 2000s, most methods adopted in infant research are related to neural networks including scaled conjugate gradient neural network, multi-layer perceptron, general regression neural network, evolutionary neural network, probabilistic neural network, neuro-fuzzy network, and Time Delay Neural network, etc. Hidden Markov model and Support Vector Machine (SVM) were also adopted in the 2000s. In the recent decade, many traditional machine learning methods, such as SVM, K-Nearest Neighbor (KNN), Gaussian Mixture Model (GMM), fuzzy classifier, logistic regression, K-means clustering, and Random Forest, are applied to pathological cry classification, cry reason classification, and cry sound detection.

In the same period, novel neural network architectures are used pervasively in industry and research. Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), CNN-RNN, Capsule Net, Reservoir Network, and neuro-fuzzy networks open a new chapter in infant cry research. This survey reviews infant cry research mainly focusing on the signal processing techniques and machine learning methods developed in the past decade. We first review typical databases used in the research, then introduce pre-processing approaches of infant cry signals, and describe a diversity of features either in time domain or in frequency domain as well as supra segmental features of infant cry signals. We focus on reviewing the state of the art methods using KNN, SVM, GMM, and CNN-based algorithms for classification and detection. We provide a list of resources for the researchers who are interested to work in this domain, and finally we make a point of the future work in this research area.

1.2. Project Objectives

The project has two main objectives:

- To classify baby cry into 4 classes namely; hunger, pain, discomfort and tired.
- To monitor baby and notify parents for activities such as awake or sleep, sleep posture, blanket information.



1.3. Project Benefits:

- In Mukhopadhyay's study, the highest classification accuracy by training a group of people to recognize some cry sounds is 33.09% while machine learning algorithm recognize the same set of data and reach 80.56% accuracy.[1]
- Accurately interpreting infants' cry sound can help parents take better care of their babies.
- Reduce the risk of SIDS.[3]
- The smart baby monitor detects harmful posture and notifies.

CH 2: LITERATURE REVIEW

2.1. Data Acquisition

As shown in Fig 1, automatic infant cry research generally involves five stages: data acquisition, pre-processing, feature extraction, feature selection, and classification. Discovering novel methods in any of the stages can help improve the performance of the final classification accuracy. The data acquisition stage includes recording the infant cry sounds and labeling. Most databases are recorded in hospitals or homes, labeled by doctors, nurses, or parents. Digital recorders are placed close to infants and are either operated on the spot to capture the cry signals one by one or left on to record the sound events around the infants for a long period of time. Infant sound is a short-term stationary signal, and it is assumed to be more stationary because of infants' lack of full control of the vocal tract. Due to the limitation of resources and sensitivity of infant cry data collection process, the total amount of infant cry database is very limited. From the previous review papers, we can see that the most commonly used database in infant cry research is Baby Chillanto database. Baby Chillanto database was collected by the National Institute of Astrophysics and Optical Electronics, CONACYT Mexico. It contains five types of cry signals including deaf, asphyxia, normal, hungry, and pain. Each cry is equally segmented into 1-s long and the total number of cries is 2268. Another database used in multiple literatures is named Dunstan Baby Language database, which is extracted from the Dunstan baby video tutorial presented by Priscilla Dunstan who invented the Dunstan Baby Language theory. There are several versions of Dunstan Baby Language database since authors extracted the audio clips in their own ways. The version in report consists of 315 wave files, sampled at 16 kHz, with a variable length between 0.3 and 1.6 s. Each utterance is a word of infant speech corresponding to one of the five "Dunstan words," which were translated as "Neh" = hungry, "Eh" = need to burp, "Oah" = tired, "Eairh" = low belly pain, and "Heh" = physical discomfort. Many databases are self-recorded for research. Researchers need to contact other authors to check availability of desired databases. One database named Donate A Cry is available online, but it is not well labeled and only one literature is found using this database. Some databases are recorded in the Neonatal Intensive Care Unit (NICU), pediatric clinics, or baby-sitting environments. Some cry audio signals online are also collected. Some synthetic databases are created by the authors in order to compare the performances of the proposed methods on real databases and synthetic

databases. In Ferretti's work, the CNN detects the cry signal better on the synthetic database than the real database. It shows that the automatic detection and classification of real-time infant cry is still challenging because the real-time environment may exist many types of complications that can affect the quality of the cry signals. Synthetic databases can be generated by adding noises to clean cry recordings or combining different cries together. Training models on synthetic databases can avoid requiring a large amount of data to be acquired in sensible environments such as NICUs. The average size is 2983 and only one database is close to 20,000 samples. Due to the sensitivity of collecting the cry data, especially pathological cry signals, small dataset size is one of the challenges in infant cry research. Data augmentation techniques are used to artificially increase the data size. Zhang et al. created new waveform images from training datasets by transforming these waveform images into slightly faster or slightly slower waveforms for the purpose of increasing training datasets to overcome over fitting problem. Several data augmentation techniques, such as noise variation, signal intensity variation, tonality variation, and spectrogram's size alteration, were used to artificially increase either the number of audio signals or the number of spectrograms. The experimental results showed that these data augmentation methods cannot lead to accuracy improvement. The reasons lie in the fact that the limited data cannot capture the diversity of variations within infant cry signals.

Figure 1- Five stages of infant cry research

2.2. Pre-Processing

The main tasks in pre-processing stage are de-noising and audio segmentation. The complication of the recording environment leads to unclean infant cry signals. In a neonatal care unit, besides infant cry signals, there could be many kinds of sounds such as footsteps, adult's speech, air-conditioner sound, alarm sound, etc. To detect or classify cry signals accurately, cleaning up the recorded data at the pre-processing stage is a crucial step. To clean up a signal, the first task is de-noising, which removes the background sounds such as speech, fan, footstep, etc. Turan and Erzin applied high-pass FIR filter to remove the speech sound and low frequency noise in the recording. Ferretti et al. reduced coherent noise source by a filter-and-sum beam former and uses OMLSA post-filter to reduce the residual diffuse noise. Gu et al. used optimized Blackman window to handle each frame signal, which is the result after the endpoint detection. The signal noise is significantly reduced after filtering. Audio segmentation task is commonly performed using Voice Activity Detection (VAD). VAD technique is widely used in speech recognition to detect the human speech in audio signals. Researchers also use it to detect the infant cry and remove the silent duration in a sample recording.

VAD also faces the challenge of separating the cry and noise. Pan et al. uses it to detect the presence or absence of baby cry in a noisy environment to improve the overall baby cry recognition rate and it is used to detect the sections of the audio with sufficient audio activity. Authors implemented a basic VAD algorithm, which uses short-time features of audio frames and a decision strategy for determining sound and silence frames. Sometimes researchers also manually cut the samples to remove the silent part and the voice interference part, and only the continuous crying part of the sound was retained.

2.3. Feature Extraction

Infant cry signal differs from adult speech. Figure 2 gives a comparison of spectrograms between infant sound and adult speech. We can see that the variations within wave form and spectrogram are quite different, especially in the areas of energy, intensity, and formants. In general, infant cry is a combination of vocalization, silence, coughing, choking, and interruptions, which includes a

diversity of acoustic and prosodic information at different levels. It is the only way for babies to communicate with the world.

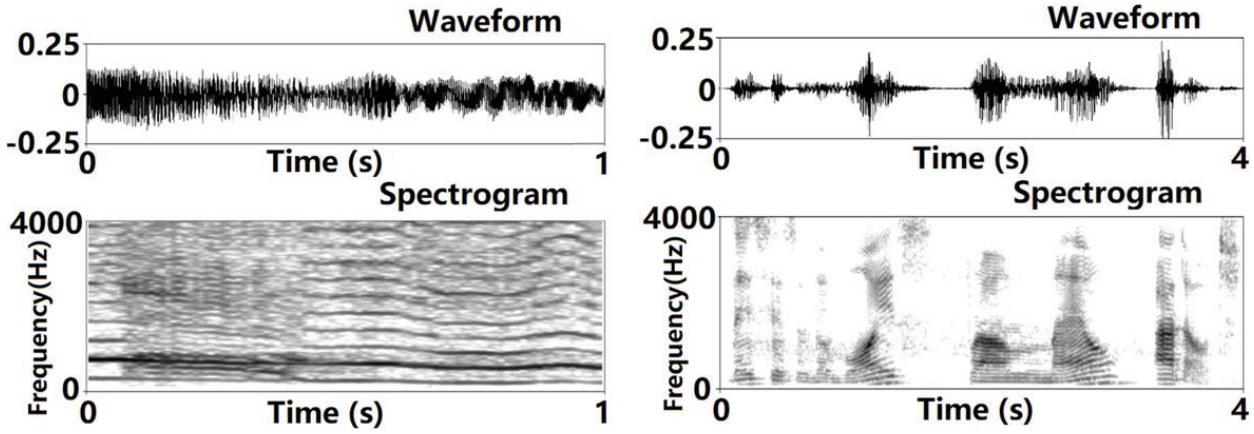


Figure 2- Adult speech vs. infant cry signal in time and frequency domain

Feature extraction is the stage to extract the discriminative features from the audio signals and later feed into the machine learning algorithms. It is one of the most vital parts of a machine learning process. Performing feature extraction task either in time or frequency domain addresses the fundamental work of baby cry analysis and processing. Time domain features, such as zero-crossing rate, amplitude, and energy-based features, etc., is simple and straightforward to compute. While time domain features are not robust enough to cover the variations within infant cry signals and the features are sensitive to background noises, the frequency domain features have strong ability to model the characteristics within infant cry signals. The commonly used MFCCs, LPCCs, and LFCCs have proven better performance than using time domain features. On the other hand, it is shown that infant cry signal is rhythmic and has cyclic changes due to the natural interruption and breath. The high-level information, such as prosodic features, are important to improve the discriminative ability within signals. Therefore, attaching prosodic domain features together with time or frequency domain is capable for capturing both physical and physiological information. In addition, spectrogram is an image that is a time-frequency representation of an audio clip. It is known that spectrogram has a strong ability to present the signal and include both acoustic and prosodic information.

Figure 3 depicts the main categories of the audio features that are applied to research related to speech, music, and environmental sounds. Acoustic and prosodic features are commonly used for infant cry detection and classification. Cepstral domain features, prosodic features, and image-based features are widely

used in speech processing and infant cry processing with a proportion over 70% research articles. In this section, we review feature extraction approaches in the latest research work.

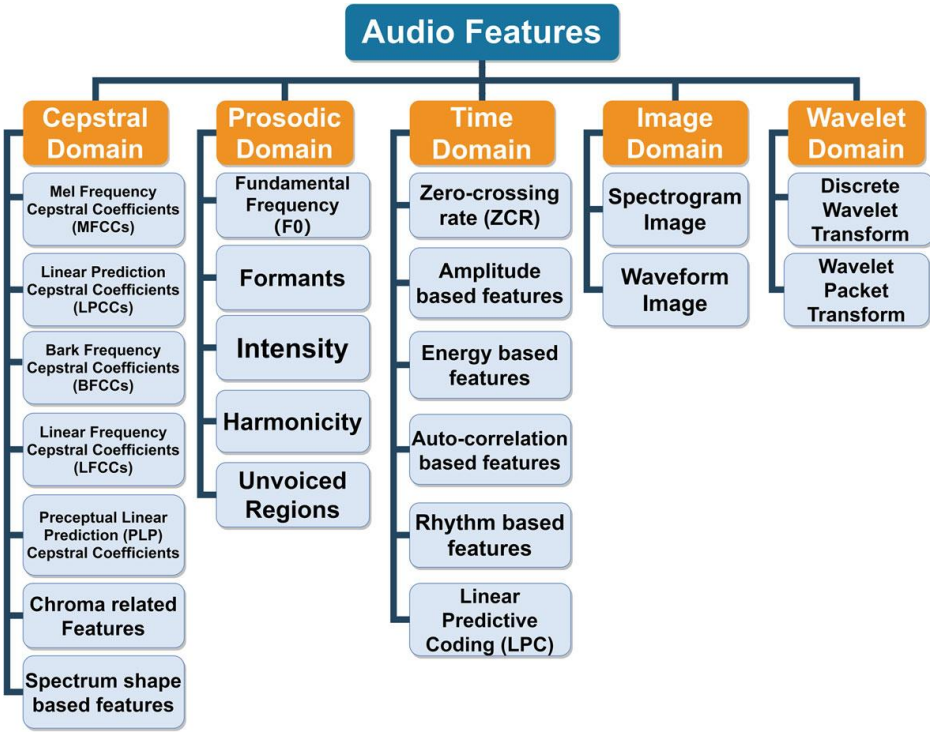


Figure 3- Main audio feature categories

2.3.1. Cepstral domain features

Mel-frequency cepstral coefficient (MFCC) is widely used in speech recognition. It is a cepstral representation of the audio signals. Researchers use it to test proposed approaches and often use it for baseline experiments. Liu et al. used MFCC along with two other cepstral features Linear Prediction Cepstral Coefficients (LPCC) and Bark Frequency Cepstral Coefficients (BFCC) for infant cry reason classification. The result showed that the BFCC with a neural network model produces the best recognition rate of 76.47%. The main idea of LPCC is to remove

the redundancy from a signal and tries to predict next values by linearly combining the previous known coefficients. It is used for cry detection. Linear Frequency Cepstral Coefficients (LFCC) extraction process is similar to MFCC extraction. The difference is that it uses a linear filter-bank instead of the Mel filter-bank. The authors showed that LFCC performs better than MFCC in discriminating high frequency audio signals such as female voice and baby cry signals. Singh et al. explored the residual MFCC and implicit LP residual features that represent excitation source information. Researchers have also tried other cepstral features such as Fast Fourier Transform (FFT), Log-Mel feature, Mel Scale, Constant-Q Chromagram , Log-mel spectrum, and delta spectrum. According to auditory perception models, MFCC coefficients are more robust than other coefficients such as LPC coefficients. In our previous work, MFCC features of normal and abnormal infant cry signals within a certain frame combined with 12 orders were plotted in a space. It is observed that the acoustic features of normal infant cry signals are quite different from the asphyxiated ones as shown in Fig 4. It indicates that the value range and tendency of acoustic features of normal and asphyxiated infant cry are different.

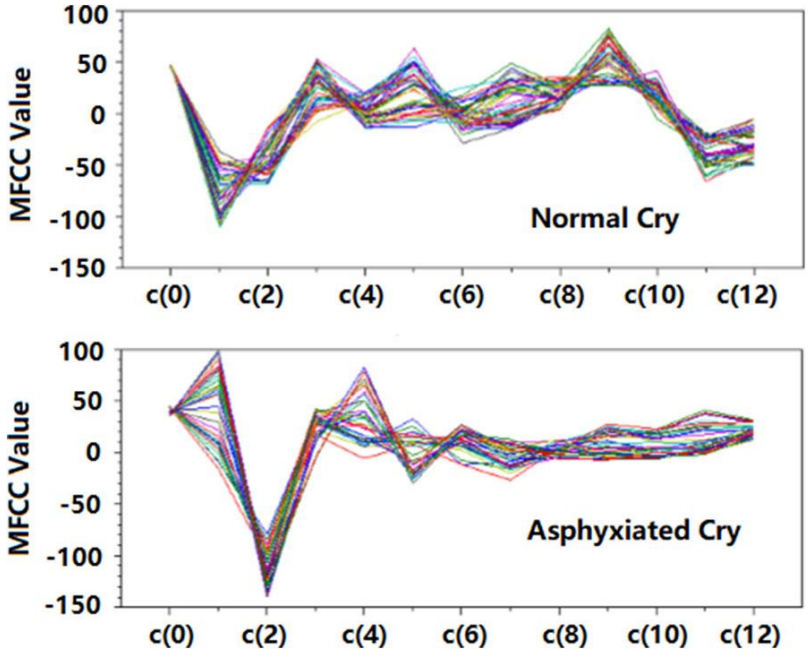


Figure 4- Multiple order MFCC features of normal and asphyxiated infant cry

2.3.2. Periodic domain features

It is shown that infant cry is made of four types of sound: one coming from the expiration phase, a brief pause, and a sound coming from the inspiration phase followed by another pause. Variations in intensity, fundamental frequency (F0), formants, and duration are typical acoustic cues that carry prosodic information about infant cry and speech. It is shown that the above prosodic features are efficient to identify the types of infant cry. Adult F0 ranges between 85 and 200 Hz while infant crying F0 is characterized by its high F0 250–700 Hz. F0 is commonly computed using an autocorrelation-based method provided by Praat.

Our previous work has shown that combining weighted prosodic features with MFCC features help improve the classification accuracy in a deep learning model. Other researchers have also found that F0 is critical in identifying infant cry signals. Chittora and Patil used F0 to calculate unvoiced segments ratio and found out unvoiced percentage in a cry is an important parameter for analysis of infant cry. Orlandi et al. used mean, median, standard deviation, and minimum and maximum of F0 and F123 to exploit differences between full-term and preterm infant cry. In 2017, Torres et al. used three handcraft features (voiced/unvoiced counter, consecutive F0, and harmonic ratio accumulation) to show comparable detection performance but resulting in 20 times lower computational cost than standard MFCCs with no additional memory cost.

2.3.3. Image domain features

Spectrogram is an image that is a time-frequency representation of an audio clip. It is known that spectrogram has a strong ability to present the signal and include both acoustic and prosodic information. Spectrogram can be extracted through framing, FFT, and calculating the log of the filtered spectrum steps illustrated in Fig 5. Feeding spectrograms into classifiers can solve the problem of different cry signals having different durations. Instead of using zero padding to achieve same length of feature vectors, normalization is applied in the process of spectrogram generation, which produces the same size images without changing the original signal. Besides feeding the spectrogram into CNN and capsule neural network, researchers take extra step to use the spectrogram image to retrieve extra features such as Local Binary Pattern (LBP), Local Phase Quantization (LPQ), and Robust Local Binary Pattern (RLBP) to help improve the classification performance.

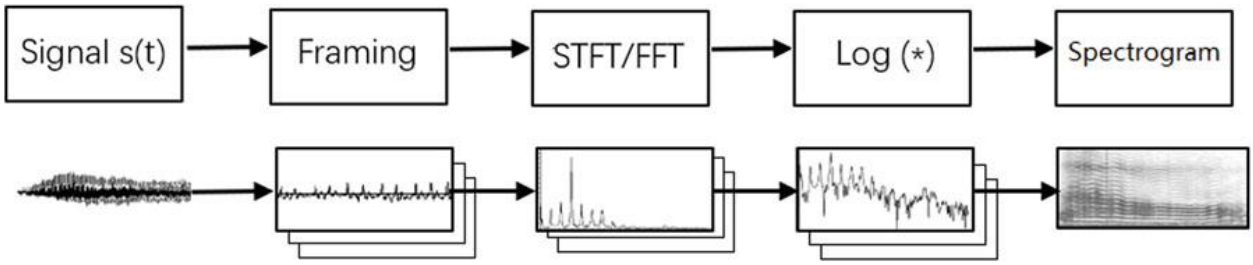


Figure 5- Flowchart of spectrogram generation

Waveform image represents the pattern of sound pressure amplitude in the time domain. It is also used in deep learning models such as Alex Net to achieve above 90% accuracy on identifying the asphyxia cry. In our previous work, we use Praat to generate images containing the prosodic feature lines including F0, intensity, and formants. The prosodic feature images CNN model is good at identifying certain types of cry signals. Combining it with spectrogram CNN and Wave form CNN produces 5% better accuracy on Baby Chillanto database and 4% on Dunstan Baby Language database.

2.3.4. Other relevant domain features

Other domain features used in infant cry research include time domain features such as zero-crossing rate, short time energy, and voiced-unvoiced regions, etc. Zero crossing rate is the rate at which the signal passes zeros and changes signs. It can be used in conjunction with short-time energy to detect endpoints of speech utterances, hence to detect the existence of the cry sound from other sounds happening in the environment. Since the amplitude of an audio signal varies with time, the short-time energy can serve to differentiate voiced and unvoiced segments. It is used for infant cry detection and classification. Torres et al. used voiced-unvoiced counter, which counts all frames having a significant periodic content, as one of the features for cry detection. Linear Predictive Coding (LPC) serves as a time domain measure of how close two different waveforms are and it is used for infant cry classification.

Wavelet Transform is a method to convert the audio signal into time-frequency domain. The waveform packet transform was used in asphyxia classification research and reached high accuracy of 99% with neural network models. It also

performs well in infant cry reason classification. The Discrete Wavelet Transform MFCC (DWTMFCC) features work well with SVM and neural network architectures. Researchers also calculate the statistical natural parameters of the data such as mean frequency, standard deviation, and third quartile range, etc. to help infant cry detection and classification. Feature extraction is a critical step in audio processing. Besides aforementioned Praat software, feature extraction tools such as Lib ROSA library and Open SMILE toolkit have made audio feature extraction easier.

2.4. Feature Selection

Feature selection is the process of selecting a subset of features from the original features extracted from the audio signals using the feature extraction techniques. The objective is to reduce the dimensionality of the features without reducing classification accuracy. Less feature require less computational resources, and hence make building smart infant cry detection and classification devices possible and affordable in the future. The original features may also contain some redundant information that prevents effectively differentiating the different types of cry signals.

Selecting the right features to fit the specific need of the task may also improve the classification accuracy. This section reviews some feature selection methods applied to the infant cry research. F-ratio method was used to select the top 20 MFCC features. The coefficients that have significant importance have higher F-ratio scores. In 2013, Yamamoto et al. used Principal Component Analysis (PCA) to reduce the dimensionality of FFT features. Forward variable Selection Method (FSM) was applied to infant cry classification by Wang in 2010 and Okada et al. proposed Iterative FSM (IFSM) based on cross-validation concept in 2011. Later, Binary Particle Swarm Optimization (BPSO) was used to remove the redundant features and keep the significant features from MFCC coefficients. Orlandi et al. used a software called Biovoice to extract 22 features from the cry signal and then used a genetic algorithm-based search method to select the best features to feed to the classifiers. In 2016, Wahid et al. compared five feature selection methods: One R, Relief F, Fast Correlation-Based Filter (FCBF), Consistency-Based Subset Evaluation (CNS), and Correlation-Based Feature Selection (CFS). It is proven that the feature selection techniques were able to greatly reduce the feature space, hence to reduce computational time. Most selection technique can also improve the performance of the neural network classifier.

In 2019, Tuduce et al. utilized three Best Feature Selection (BFS) approaches to exclude irrelevant features and redundant features and tested them with 35 classifiers. The feature set is reduced from over 6000 features to 500 and the result shows that BFS can improve the classification accuracy for some classifiers. Feature selection techniques remove the features irrelevant to the specific task, so it can reduce the feature space, save computational time, and improve classification accuracy.

2.5. Infant Cry Classification

With data cleaned and segmented and features extracted, selected, and normalized, finding the appropriate classifier is the most important stage in the machine learning process. In this section, we review some popular machine learning methods and applications used in infant cry classification in the past decade.

2.5.1 Infant cry classification models

2.5.1.1 Traditional machine learning classifiers

A) Support Vector Machine

The most popular probabilistic classifier used in infant cry classification is Support Vector Machine (SVM). The types of SVM include multiclass SVMs, linear, and RBF kernels binary SVM. The features fed into the SVM include temporal features, prosodic features, and cepstral features. In 2017, Onu et al. compared SVM to other non linear classifiers like neural networks on asphyxia classification and concluded that SVMs are designed to work effectively with limited examples and high dimensional data. In 2015, Chang et al. used the incremental SVM learning model, which keeps adding new data into the dataset in each training step, producing more than 18% better accuracy than the original SVM model on infant cry classification based on FFT features.

B) K-Nearest Neighbor

KNN is a well-known pattern recognition method used in classification. There are k nearest neighbors in the feature space. The goal is to assign test sample to the class that its nearest neighbor belongs to. If k is greater than 1, the nearest neighbor is selected based on the number of nearest neighbors. In the case of infant cry classification, researchers used Euclidean distance, Minkowski distance, and other methods to measure the distance between two sample feature vectors. Feature vectors selected are usually MFCC and LFCC. Cohen and Lavner used KNN algorithm, in which each frame is classified either as a cry or not a cry, and then the sample is classified to be cry signal if more frames in the sample is identified as cry.

C) Gaussian mixture model

GMM is a probabilistic model that assumes the data points are in Gaussian distribution of some mean and variance. The idea is to learn the parameters to model the provided training data as mixture of several Gaussian distributions. Then the test data can be classified by the trained model. Expectation Maximization (EM) algorithm is used for finding the maximum likelihood estimates of the parameters under GMM-based structures. In 2016, Banica et al. used GMM-UBM method to classify Dunstan baby cries. The universal background model (UBM) is a GMM model that is trained on large amount of general cry signals with no specific labels. The classification accuracy of the GMM-UBM with MFCC achieved 70% on Dunstan baby cries and 50.6% on SPLANN database. GMM-UBM is also used by Alaie et al. to classify healthy cries and pathological cries. The Boosting Mixture Learning adaptation method proposed outperforms the MAP algorithm. In 2019, Sharma et al. compared the GMM clustering to hierarchical clustering and K-means clustering on cry features and showed the GMM model produces the best result with least amount of overlapping data points with a certain database. It is shown that GMM-based classifiers are sensitive to environment and cannot lead to satisfied results especially with limited training data.

D) Fuzzy classifier

Fuzzy logic systems have been used in many applications such as transmission systems, power systems, and wireless network routing. It is also used in infant cry classification. Selected features are converted into fuzzy values in the fuzzification step, certain fuzzy membership functions are used, and fuzzy rules are defined. Kia

et al. used fuzzy classification to detect infant cry signals from laughter signals. Rosales-Pérez et al. used fuzzy decision tree, fuzzy decision forest, fuzzy KNN, and fuzzy relational neural network classifier for pathological cry classification. Type-2 fuzzy pattern matching algorithm is used to classify asphyxia, normal, and hyperbilirubinemia. It also outperforms SVM and logistic regression classifier on classifying hunger and pain.

E) Logistic regression classifier

Logistic regression classifier is a low-complexity supervised algorithm, and it is usually used as a referencing experiment for infant cry research. Lavner et al. used it to show that CNN performs better on cry detection and Orlandi et al. used it to compare with many other classifiers, in which random forest performed the best on classifying full-term and preterm infant cries.

F) K-means clustering

K-mean clustering represents an unsupervised algorithm mainly used for clustering. Unlabeled data points can be gradually separated into groups based on the mean value and centroid moving. Sharma et al. used K-means clustering to show that the GMM model has better performance differentiating different types of cry. K-means clustering was used to build a speaker database for speaker recognition.

G) Bagging, boosted trees, and random forest

Bagging, boosted trees, and random Forest are techniques that perform ensemble decision trees. They all combine multiple decision trees to produce better performance. Experiments have shown that they are powerful on infant cry classification. Osmani et al. showed bagging and boosted trees outperform SVM. Milano et al. compared it to MLP, SVM, Reservoir Network, GMM, and HMM models and showed random forest classifier is next to Reservoir Network. An open source data mining software named Waikato Environment for Knowledge Analysis (WEKA) is used. Among over 100 classification algorithms implemented in WEKA, random forest outperforms SVM, MLP, logistic regression, and BayesNet, etc. Tuduce et al. tested 40 classifiers in WEKA and the tree classifiers showed the best overall performance comparing to Bayes classifiers, lazy classifiers, function classifiers, and rule classifiers, etc.

2.1.5.2. Neural network-based models

Artificial Neural Network (ANN) is a machine learning method. In 1995, Petroni et al. made the first attempt of ANN in infant cry classification.

A) Feed Forward Neural Network (FFNN)

It is the simplest neural network and Multi-Layer Perceptron (MLP) is a type of FFNN that contains at least three layers. The experiments showed that FFNN's performance was not as good as nearest neighbor classifier based on MFCC features. MLP was used in with MFCC for identifying pathological cries. To classify asphyxia, Hariharan et al. used Probabilistic Neural Network (PNN), General Regression Neural Network (GRNN), and Time-Delay Neural Network (TDNN) and achieved above 97% accuracy.

B) Convolutional Neural Network

It is a deep learning algorithm that has been successfully used in computer vision, language processing, and other domains achieving unprecedented high accuracy. Multi-channel CNN, which accepts multiple channel input, were applied in Manikanta et al. used 1D CNN on MFCC features for cry detection and the result outperformed feed forward neural network and SVM classifier. In 2019, Le et al. applied transfer learning with CNN on spectrograms on Baby Chillanto database and achieved promising result.

C) Long Short-Term Memory (LSTM)

It is a type of Recurrent Neural Network (RNN), which has internal states to make accepting sequence of data possible and is known as best neural network for time series data such as language translation and speech recognition. Mark Huckvale fed the low-level signal temporal features into the bidirectional LSTM model and later combine with another two dense-layer neural network in. The LSTM itself and combined network both outperformed the baseline SVM model.

D) CNN-RNN

It is a deep learning architecture combining CNN with RNN. It has shown its power in sound detection and classification. In Detection and Classification of

Acoustic Scenes and Events (DCASE) 2017 competition, Lim et al. used it to win the first place on detecting the target sounds (baby crying, glass breaking, gunshot) with mixed noisy background. In 2019, Maghfira et al. used CNN-RNN to classify the five types of cries and reached the highest accuracy of 94.97% for Dunstan Baby Languages database.

E) Neuro-fuzzy Network

It combines fuzzy logic with neural networks and it has been used successfully by researchers in infant classification. In 2009, Santiago-Sánchez et al. used type-2 fuzzy set to classify asphyxia and hyperbilirubinemia. In 2012, Molaeezadeh et al. proposed a type-2 fuzzy pattern matching classifier and it outperformed SVM and logistic regression classifiers in classifying hunger and pain. In recent years, it is noticed that combining fuzzy systems with neural networks can unite their advantages and evade the disadvantages of both methods. Fuzzy systems require rules while neural networks directly learn from data. Neuro-fuzzy approach was used to classify Dunstan baby language type of cries. Neural networks were trained and the Mandani fuzzy logic was adopted after data normalization to create new "transformed dataset," which is used for final classification step of KNN. The classification accuracy reached 86.25%, which is better than normal neural network model, SVM, and GMM methods.

F) Capsule Network

It is a deep learning topology adds a structure called capsules into the CNN model. As max pooling in CNN only picks the maximum value within a region and throws away information in certain positions, higher-level capsules cover larger regions of the image and performs routing by agreement instead. CapsNet was applied to classify infants' emotional cry in domestic environments and the accuracy is improved more than 10% over the CNN model with spectrograms.

G) Reservoir Network (RN)

It is a neural network model derived from RNN. Its input nodes connect to a non trainable reservoir, which contains connected nonlinear units with randomly generated fixed weights. Ntalampiras used RN in infant cry multi-class classification with fused feature sets and showed that RN model outperformed MLP, SVM, random forest, GMM clustering, etc.

Many machine learning methods have been experimented in infant research. Each of them has advantages and disadvantages and no algorithm is the perfect for every dataset and task. Selecting a suitable model to achieve high performance is challenging. To determine the classification ability of the different models, Fuhr et al. experimented differentiating healthy infant cries and cries of infants suffering from several diseases using 12 classifiers including SVM, decision tree, KNN, MLP, etc. The result showed only C5 decision tree and KNN achieved greater than 90% accuracy. Applying many algorithms on the task before selecting the algorithm to use is impractical. Comparing the machine learning algorithms used in infant research, we analyze them from the following aspects. Readers can choose the appropriate algorithm accordingly for their datasets and tasks.

• **Time complexity**. It includes training time and classification time relying on the data size, searching space, and the complexity of coefficients. In general, traditional methods such as SVM, K-means clustering, and GMM-based approaches are relatively simple and straightforward. Smaller sample size is acceptable, which differs from neural network methods. Hence, training time, searching time, and classification time are much less than those of neural network methods. Also, fine tuning in neural network models also requires more developing time.

• **Sample complexity**. It indicates whether the model requires large size of data or not to learn. It depends on the complexity of the data and the complexity of the algorithms. To reach better performance, neural network methods generally require larger sample size for complex searching space than other traditional algorithms. Larger size infant cry databases are needed for deep neural networks.

• **Parametricity.** It indicates if the number of the parameters used in the model is fixed or it varies along when new data is brought in. Linear regression, GMM, and neural networks are parametric methods while KNN and SVM are nonparametric models.

• **Feature complexity.** Features extracted from either time domain or frequency domain have the same abilities to represent the different characteristics of the cry signals in different models. There is no feature complexity difference involved for traditional models or neural network-based models. But using too many features to

represent one sample may cause over fitting issue; therefore, selecting the most appropriate features for specific models is critical.

• **Parallelizability.** Parallelizability is a pivotal feature for saving the training time of machine learning methods. Large amount of data in neural networks is associated with high computation cost in both time and space. Parallelizability with Graphics Processing Unit (GPU) computing greatly reduces the training time and made deep learning possible. Other method such as KNN is easy to parallel, but parallelism is tricky if the next step is based on the previous step result such as decision trees.

2.5.2. Infant cry applications

Researchers use different classifiers to perform infant cry processing tasks. In the past decade, most research work continue to pay effort to improve the classification accuracy of infant cry signals including differentiating the pathological cries from the normal cries and understand the meaning behind the cry signals. In this section, we review the significant works on infant cry classification and detection.

2.5.2.1. Infant cry reason classification

In the early years of infant cry research, more works were performed on automatically differentiating the cries of healthy infants from pathological cries. In recent years, exploring the meaning of the cries attract more research interests. It is noticeable that researchers are using different datasets, most of which are self-recorded. With different datasets in similar research, even the classification types are the same, it is unfair to make direct comparison on the performances of the proposed methods. The infant classification remains in challenging stage due to the lack of standard public datasets and the classification accuracy is still relatively low.

2.5.2.2. Infant pathological cry classification

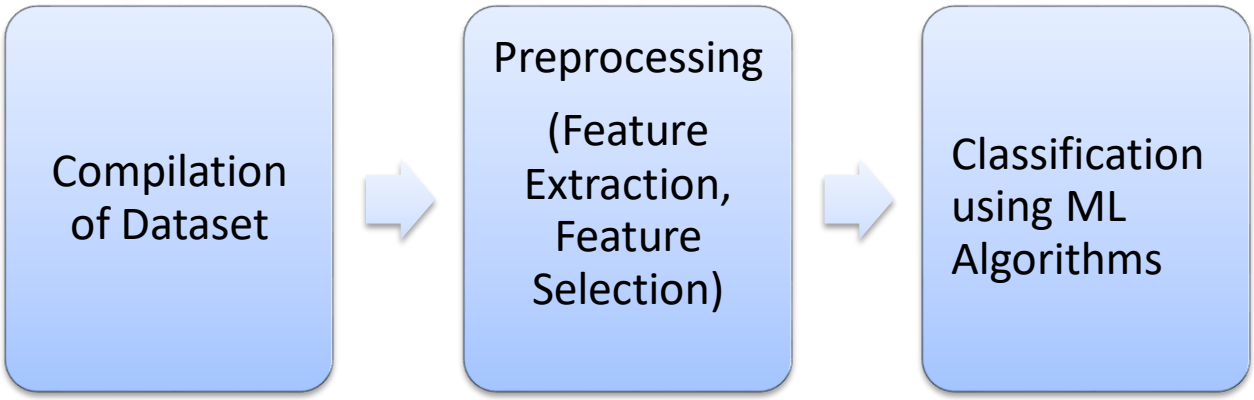
Infant cry signals have been used to identify many diseases such as asphyxia, hypo-acoustic (hearing disorder), hypothyroidism, hyperbilirubinemia, cleft palate, respiratory distress syndrome, ankyloglossia with deviation of the epiglottis and larynx, etc. Readers can find the related works on pathological cry classification before 2011 in. In the past decade, researchers continue to apply novel methods to classify normal cry and pathological cry. Asphyxia cry is the most popular disease in research. Researchers have been using the Baby Chillanto database to perform the binary classification. In 2012, Probabilistic Neural Network (PNN) and General Regression Neural Network (GRNN) reached 99% accuracy, the latest SVM model reached 97.7% accuracy, and the deep learning FFNN model reaches 96.74% accuracy.

Author	Dataset	Classes	Performance	Features Used
Felipe(2019) [4]	**iCOPE**	**2 (Pain VS No Pain)**	**71.68%**	**MFCCs**
Chang (2016) [5]	**Collected from National Taiwan University Hospital**	**3 (hungry ,pain, and sleep)**	**78.5%**	**Spectrograms**
Yamamoto (2013) [6]	**Self – recorded**	**3 (discomfort, hungry, sleepy)**	**62.1%**	**FFT**
Sharma [7] (2019)	**Donate a Cry**	**3(Hungry ,belly-pain, burp needed)**	**81.27%**	**Spectral entropy**
Liu (2018) [8]	**NICU recorded**	**3 (Draw-attention, diaper change, hungry)**	**76.4%**	**MFCC**

Company/Work	Features	Cost
NANIT	**Sleep tracking and detection, live feed.**	**300$ and monthly subscription**
Motorola Connect view	**Live feed, Lullaby**	**249$**
Miku Pro	**Sleep monitoring, temperature detection, lullaby.**	**299$**

CH 3: METHODOLOGY

3.1. Flowchart



- Baby's cry will be recorded using a cell phone and will be classified into one of the 4 classes using ML algorithms.

• 3.2. EAR Ratio and Pose Estimation

- 'Donate a cry' dataset will be used along with our own collected data.
- Baby's sleep/awake state will be detected using EAR ratio or pose estimation.

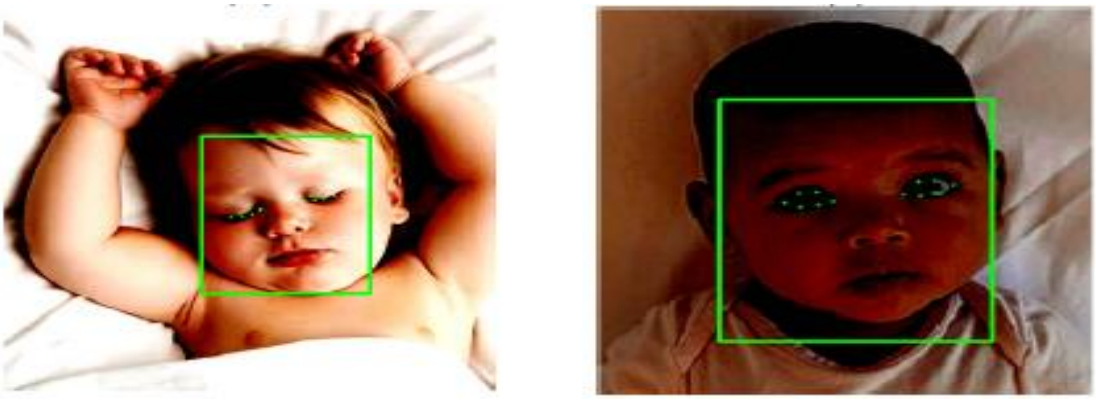


Figure 6- Ear Ratio

Figure 7- Pose Estimation

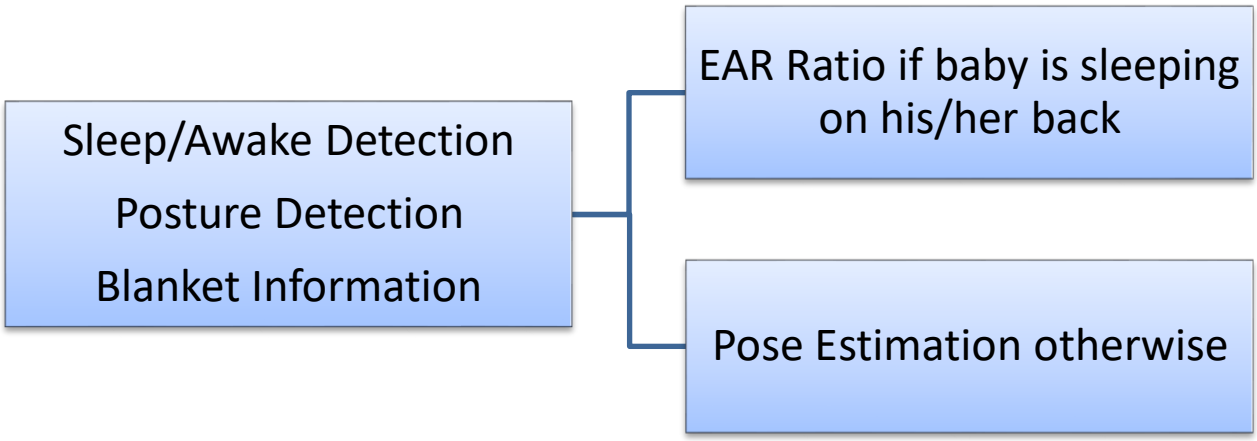
3.3. Components for sleep/awake detection

- Night vision camera and a microcontroller will be used for sleep/awake detection.

Figure 8- Night Vision Camera

Figure 9- Jetson Nano Microcontroller

3.4. Design Workflow



CH 4: SOFTWARE

4.1. Project Work

The project comprises of 2 parts:

- Firstly, we classified baby's cry into 4 classes namely; hunger, discomfort, tiredness and belly pain. Cry of a baby would be recorded via mobile app.
- Second part of the project was regarding baby monitoring. Camera and raspberry pi was used to monitor the baby, results would be shown on the app such as if the baby is sleeping or awake.

4.2. Implementation of Cry Classification

- Librosa library in python was used for feature extraction.
- Feature selection was done using correlation and information gain.
- Gradient Boosting Classifier was used for classification.

4.2.1. Pose Estimation

- Estimating the spatial locations of key body points.
- It was done using media pipeline in open cv.

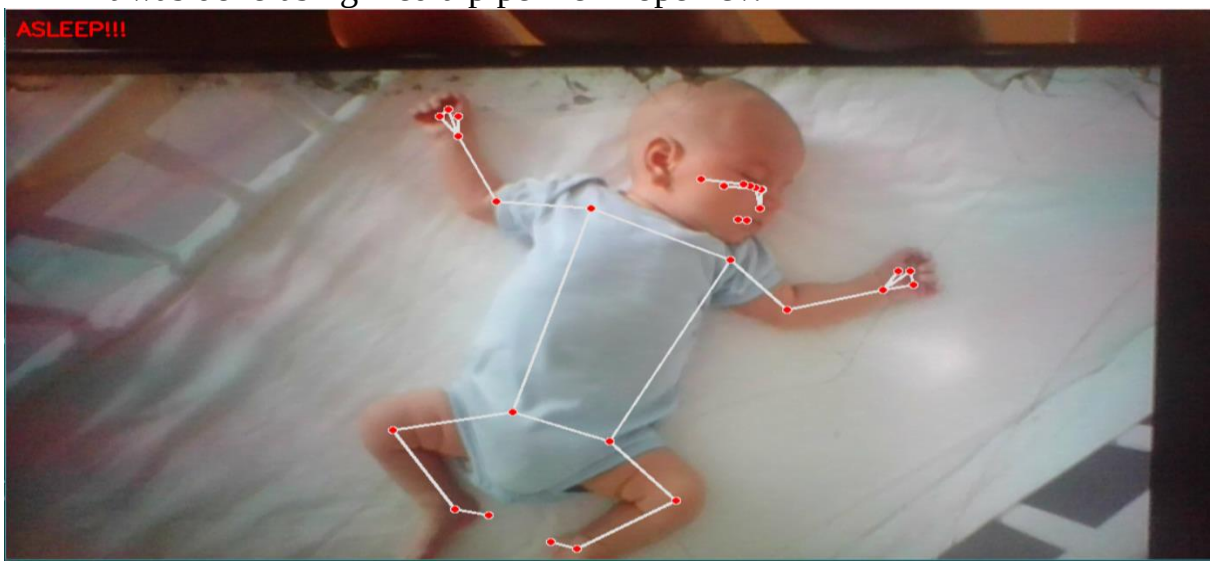


Figure 10- Pose Estimation

4.2.2. Eye Aspect Ratio (EAR)

- EAR finds the horizontal and vertical distance of the eye coordinates.
- The ratio determines whether the baby is asleep or awake.

$$EAR' = (BF + CE)/2AD$$

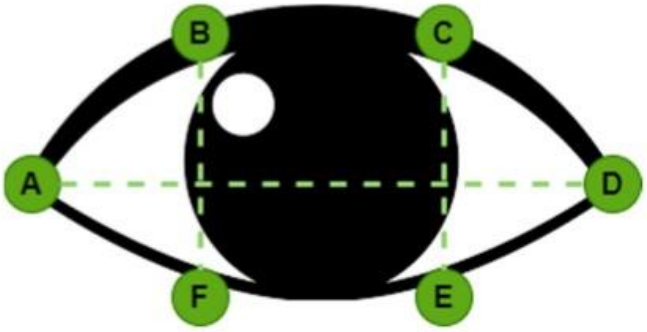
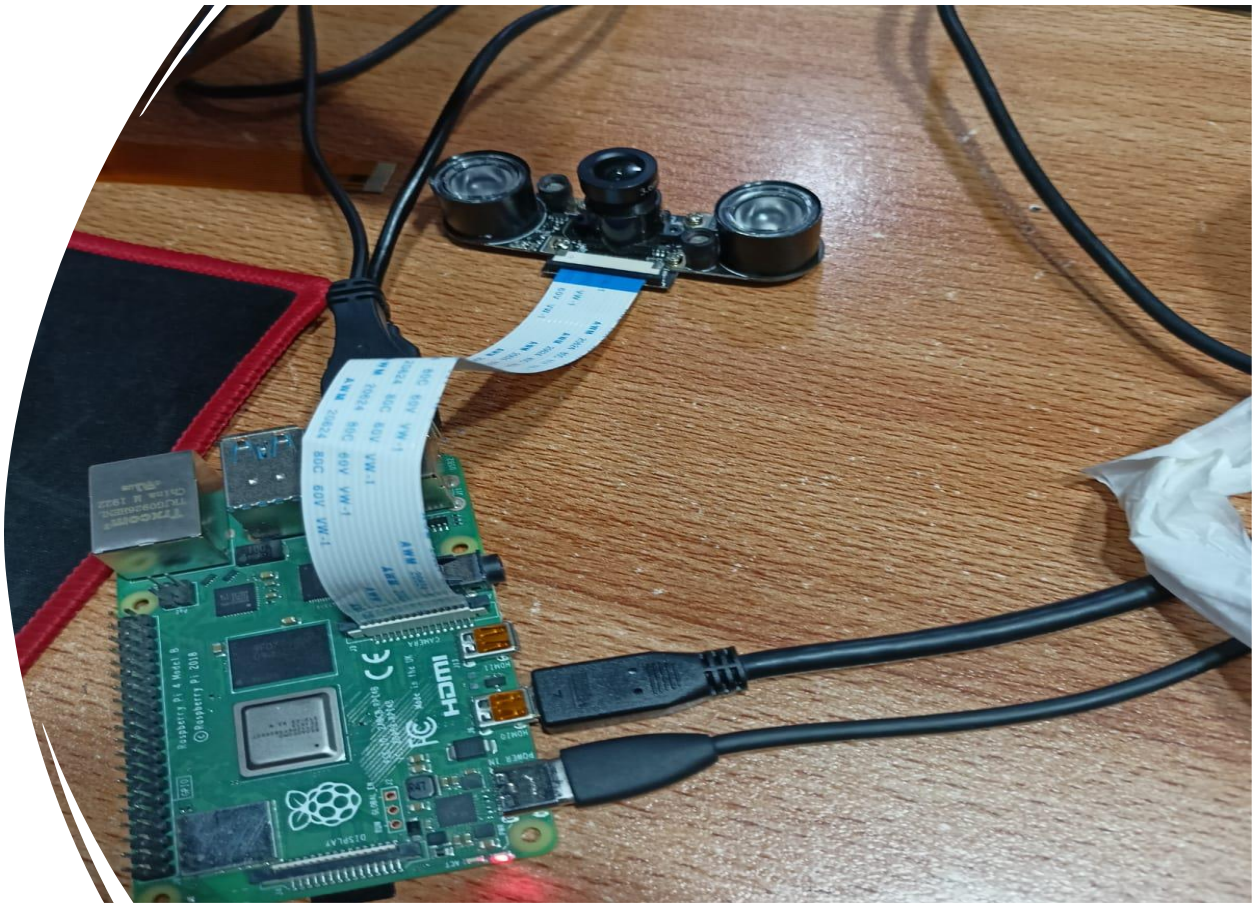


Figure 11- Eye Aspect Ratio

CH 5: HARDWARE

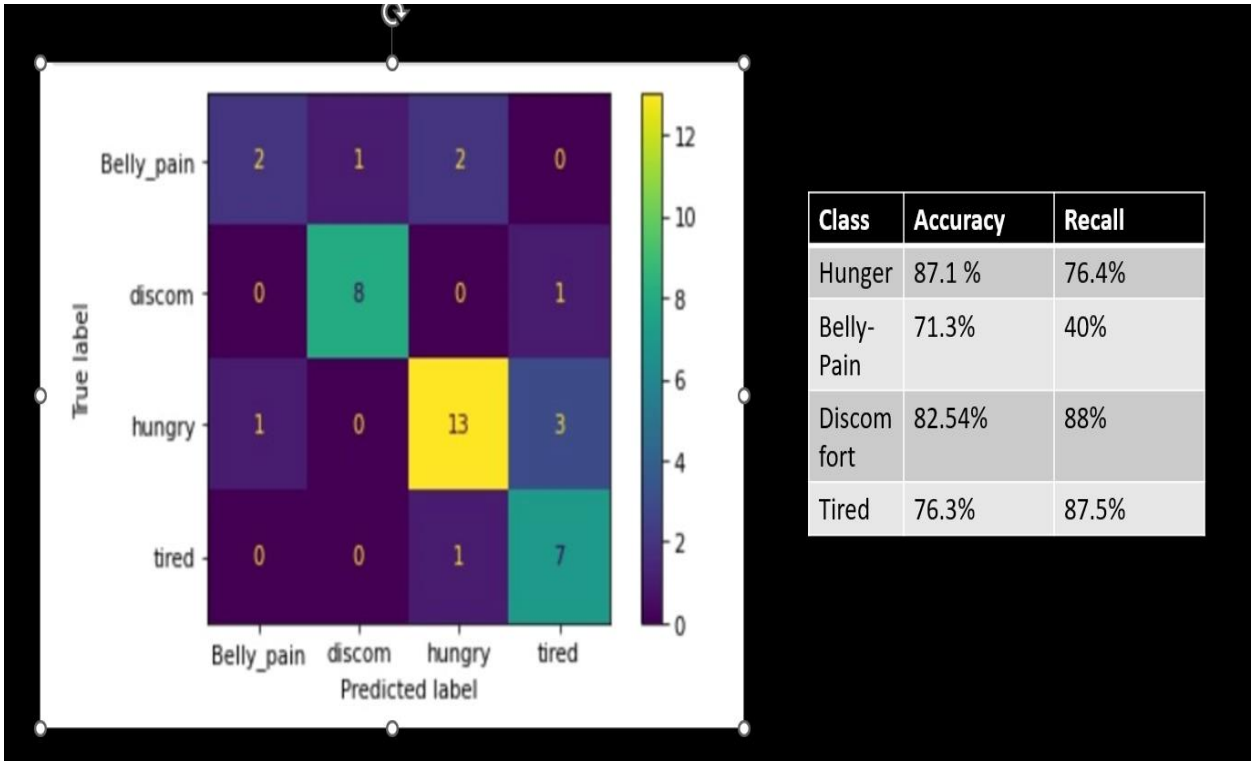
5.1. Development

- Implementation of the whole project on Raspberry pi 4.
- Used Pi camera module along with Logitech webcam.



CH 6: RESULTS AND DISCUSSIONS

6.1. Confusion Matrix



Class	Accuracy	Recall
Hunger	87.1 %	76.4%
Belly-Pain	71.3%	40%
Discomfort	82.54%	88%
Tired	76.3%	87.5%

6.2. Algorithms

Model	Accuracy
Logistic Regression	38.4%
Gaussian NB	41%
SVC	43.6%
K neighbors Classifier	48.7%
Gradient Boosting Classifier	76.9%

6.3. Suggestions for Future Work

- Increase the accuracy for cry classification.
- Record Sleep/Awake patterns using database.
- Blanket information using pose estimation.
- Cry detection and play lullaby.

CONCLUSION

In this project report, we describe the significant research work in infant cry analysis and classification, providing details and resources that are helpful for both researchers and medical professionals who work in this area. It is shown that the limited database resources hinder the development of the infant cry research. Large databases with diverse samples fitting the need of deep neural networks is imperatively desired. The current tendency for feature extraction is to generate a mixed feature set and takes advantages of different domains to achieve better discriminating ability. The relevant research results show promising improvement with combined features. In addition, new neural network-based architectures become the mainstream methods. It proves better robustness and performance than traditional machine learning approaches. In the future, we are interested in creating a large database, extracting more robust features, combining features with good ratio, establishing novel neural network architectures with the use of prior knowledge as well as other space information from interdisciplinary areas.

REFERENCES

- [1] An evaluation of human perception for neonatal cry using a database of cry and underlying cause, (2013). https://doi.org/10.1109/IndianCMIT.2013.6529410 [1]

- Nordqvist, C. New Parents Have 6 Months Sleep Deficit During First 24 Months of Baby's Life. Medical News Today, 25 July 2010. Available online: https://www.medicalnewstoday.com/articles/195821.php/ [2]

- The Bump. Is It Okay for Babies to Sleep on Their Stomach? Available online: https://www.thebump.com/a/baby-sleeps-on-stomach/ [3]

- G. Z. Felipe, R. L. Aguiat, Y. M. G. Costa, C. N. Silla, S. Brahnam, L. Nanni, S. [4]

- Application of deep learning for recognizing infant cries, (2016), pp. 1–2. https://doi.org/10.1109/ICCE-TW.2016.7520947 [5]

- Recognition of a baby's emotional cry towards robotics baby caregiver. Int. J. Adv.Robot. Syst. 10 (2013). https://doi.org/10.5772/55406 [6]

- Infant weeping calls decoder using statistical feature extraction and Gaussian mixture models, (2019), pp. 1–6. https://doi.org/10.1109/ICCCNT45670.2019.8944527 [7]

- Infant cry signal detection, pattern extraction and recognition, (2018), pp. 159–163. https://doi.org/10.1109/INFOCT.2018.8356861 [8]

APPENDIX

Codes:

1. Cry Classification:

```
import librosa
import librosa.display
import IPython.display as ipd
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.decomposition import PCA
import numpy as np
from sklearn import preprocessing
!pip install python_speech_features
import python_speech_features as mf
import pandas as pd
import os
import librosa

audio_dataset_path=r"D:\FYP\FYP_class\FULLDATA"
metadata=pd.read_csv(r"D:\FYP\FYP_class\fulldata.csv",names =
['file','fold','label'])

def feature_1(file):
    audio,sr = librosa.load(file)
    zcr = librosa.zero_crossings(audio)
    zcr = sum(zcr)
    data = pd.DataFrame([zcr],columns=['A'])
    return data

def feature_2(file):
    audio,sr = librosa.load(file)

    #mfcc=np.mean(librosa.feature.mfcc(audio,sr =
sr,n_mfcc=12).T,axis=0)
```

```
    mfcc_feature =np.mean(mf.mfcc(audio,sr, 0.025, 0.01,12,nfft
= 1200, appendEnergy = True),axis = 0)
    #mfcc_feature = preprocessing.scale(mfcc_feature)
    mfcc = pd.DataFrame(mfcc_feature)
    #df1 = pd.DataFrame(mfcc)
    #df1 = df1.T
    mfcc = mfcc.T
    return mfcc

def feature_3(file):
    audio,sr = librosa.load(file)
    chromagram =np.mean(librosa.feature.chroma_stft(audio,
sr=sr, hop_length=512),axis=1)



    cr = pd.DataFrame(chromagram)
    #df1 = pd.DataFrame(mfcc)
    #df1 = df1.T
    cr = cr.T
    return cr

data_1 = pd.DataFrame()
data_2 = pd.DataFrame()
data_3 = pd.DataFrame()
#parser function works!!!!!
import numpy as np
from tqdm import tqdm
### Now we iterate through every audio file and extract features
### using Mel-Frequency Cepstral Coefficients

for index_num,row in tqdm(metadata.iterrows()):
    file_name =
os.path.join(os.path.abspath(audio_dataset_path),'fold'+str(row[
"fold"])+'/',str(row["file"]))
    temp_1 = feature_1(file_name)
    temp_2 = feature_2(file_name)
    temp_3 = feature_3(file_name)
    data_1 = data_1.append(temp_1)
    data_2 = data_2.append(temp_2)
    data_3 = data_3.append(temp_3)

result = pd.concat([ data_1,data_2,data_3], axis=1,
ignore_index=True)
result.head(-10)
result.to_csv('file.csv')
```

```
metadata = pd.read_csv(r"D:\FYP\FYP_class\fulldata.csv",names =
['file','fold','label'])

label = metadata['label']

result_final = pd.concat([result,label], axis=1)

from sklearn.feature_selection import mutual_info_classif
# determine the mutual information
mutual_info = mutual_info_classif(result,label)



mutual_info = pd.Series(mutual_info)
mutual_info.index = result.columns
mutual_info.sort_values(ascending=False)

x_train,x_test,y_train,y_test =
train_test_split(result[[0,3,24,23,10,6,16]],label,test_size=0.2
,random_state=0)


from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

model =SVC
model.fit(x_train,y_train)

model.score(x_test,y_test)

from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(model,x_test,y_test)
y_pred = model.predict(x_test)
```

2. Pose Estimation:

```
import math
import cv2
import numpy as np
from time import time
import mediapipe as mp
import matplotlib.pyplot as plt


# Initializing mediapipe pose class.
mp_pose = mp.solutions.pose

# Setting up the Pose function.
pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.3,
model_complexity=2)

# Initializing mediapipe drawing class, useful for annotation.
mp_drawing = mp.solutions.drawing_utils




def detectPose(image, pose, display=True):
    '''
    This function performs pose detection on an image.
    Args:
        image: The input image with a prominent person whose pose landmarks needs to be
detected.
        pose: The pose setup function required to perform the pose detection.
        display: A boolean value that is if set to true the function displays the original input image,
the resultant image,
                and the pose landmarks in 3D plot and returns nothing.
    Returns:
        output_image: The input image with the detected pose landmarks drawn.
        landmarks: A list of detected landmarks converted into their original scale.
    '''

    # Create a copy of the input image.
    output_image = image.copy()

    # Convert the image from BGR into RGB format.
    imageRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
    # Perform the Pose Detection.
    results = pose.process(imageRGB)

    # Retrieve the height and width of the input image.
    height, width, _ = image.shape

    # Initialize a list to store the detected landmarks.
    landmarks = []

    # Check if any landmarks are detected.
    if results.pose_landmarks:

        # Draw Pose landmarks on the output image.
        mp_drawing.draw_landmarks(image=output_image, landmark_list=results.pose_landmarks,
                                  connections=mp_pose.POSE_CONNECTIONS)

        # Iterate over the detected landmarks.
        for landmark in results.pose_landmarks.landmark:
            # Append the landmark into the list.
            landmarks.append((int(landmark.x * width), int(landmark.y * height),
                              (landmark.z * width),(landmark.visibility)))

    # Check if the original input image and the resultant image are specified to be displayed.
    if display:

        # Display the original input image and the resultant image.
        plt.figure(figsize=[22, 22])
        plt.subplot(121);
        plt.imshow(image[:, :, ::-1]);
        plt.title("Original Image");
        plt.axis('off');
        plt.subplot(122);
        plt.imshow(output_image[:, :, ::-1]);
        plt.title("Output Image");
        plt.axis('off');

        # Also Plot the Pose landmarks in 3D.
        mp_drawing.plot_landmarks(results.pose_world_landmarks,
mp_pose.POSE_CONNECTIONS)

    # Otherwise
    else:

        # Return the output image and the found landmarks.
        return output_image, landmarks
```

```
# Setup Pose function for video.
pose_video = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.5,
model_complexity=1)

# Initialize the VideoCapture object to read from the webcam.
video = cv2.VideoCapture(0)

# Create named window for resizing purposes
cv2.namedWindow('Pose Detection', cv2.WINDOW_NORMAL)

# Initialize the VideoCapture object to read from a video stored in the disk.
# video = cv2.VideoCapture('media/running.mp4')

# Set video camera size
video.set(3, 1280)
video.set(4, 960)

# Initialize a variable to store the time of the previous frame.
time1 = 0
test=[]
temp=0
curr=0
flag=0
flag_a=0
flag_check=30
flag_check_a=20
# Iterate until the video is accessed successfully.
while (True):
    temp=curr



    # Read a frame.
    ok, frame = video.read()

    # Check if frame is not read properly.
    if not ok:
        # Break the loop.
        break

    # Flip the frame horizontally for natural (selfie-view) visualization.
    frame = cv2.flip(frame, 1)

    # Get the width and height of the frame
```

```
    frame_height, frame_width, _ = frame.shape

    # Resize the frame while keeping the aspect ratio.
    frame = cv2.resize(frame, (int(frame_width * (640 / frame_height)), 640))

    # Perform Pose landmark detection.
    frame,land = detectPose(frame, pose_video, display=False)

    # Set the time for this frame to the current time.
    time2 = time()

    # Check if the difference between the previous and this frame time > 0 to avoid division by
zero.
    if (time2 - time1) > 0:
        # Calculate the number of frames per second.
        frames_per_second = 1.0 / (time2 - time1)

        # Write the calculated number of frames per second on the frame.
        #cv2.putText(frame, 'FPS: {}'.format(int(frames_per_second)), (10, 30),
cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0),
                    # 3)

    # Update the previous frame time to this frame time.
    # As this frame will become previous frame in next iteration.
    time1 = time2


    test.append(land[0][0])
    curr=land[0][0]

    thres=30
    store=abs(curr-temp)



    if store<thres:
        flag += 1
        print(flag)



        if flag>= flag_check:
            cv2.putText(frame, "ASLEEP!!!", (10, 30),
                                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

```
    else:
        flag=0
        cv2.putText(frame, "Awake!!!", (10, 30),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)




    # Display the frame.
    cv2.imshow('Pose Detection', frame)

   #print(land[mp_pose.PoseLandmark.LEFT_ELBOW.value][0][0],
    #land[mp_pose.PoseLandmark.RIGHT_ELBOW.value][0][0],
    #land[mp_pose.PoseLandmark.LEFT_KNEE.value][0][0],
    #land[mp_pose.PoseLandmark.RIGHT_KNEE.value][0][0])



    #a1=land[mp_pose.PoseLandmark.LEFT_ELBOW.value]
    #b1=land[mp_pose.PoseLandmark.RIGHT_ELBOW.value]
    #c1=land[mp_pose.PoseLandmark.LEFT_KNEE.value]
    #d1=land[mp_pose.PoseLandmark.RIGHT_KNEE.value]
    # Wait until a key is pressed.
    # Retreive the ASCII code of the key pressed
    k = cv2.waitKey(60) & 0xFF



    # Check if 'ESC' is pressed.
    if (k == 27):
        # Break the loop.
        break



# Release the VideoCapture object.
video.release()
```

```
# Close the windows.
cv2.destroyAllWindows()
```

3. Eye Aspect Ratio:

```
from scipy.spatial import distance
from imutils import face_utils
import imutils
import dlib
import cv2

def eye_aspect_ratio(eye):
        A = distance.euclidean(eye[1], eye[5])
        B = distance.euclidean(eye[2], eye[4])
        C = distance.euclidean(eye[0], eye[3])
        ear = (A + B) / (2.0 * C)
        return ear

thresh = 0.25
frame_check = 20
detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor("models/shape_predictor_68_face_landmarks.dat")# Dat file is
the crux of the code

(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
cap=cv2.VideoCapture(0)
flag=0
while True:
        ret, frame=cap.read()
        frame = cv2.flip(frame, 1)
        frame = imutils.resize(frame, width=700)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        subjects = detect(gray, 0)

        for subject in subjects:
                shape = predict(gray, subject)
                shape = face_utils.shape_to_np(shape)#converting to NumPy Array
                leftEye = shape[lStart:lEnd]
```

```
                rightEye = shape[rStart:rEnd]

                leftEAR = eye_aspect_ratio(leftEye)
                rightEAR = eye_aspect_ratio(rightEye)
                ear = (leftEAR + rightEAR) / 2.0
                leftEyeHull = cv2.convexHull(leftEye)
                rightEyeHull = cv2.convexHull(rightEye)
                cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
                cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)



                if ear>0:
                        print('EAR')
                else:
                        continue



                if ear < thresh:
                        flag += 1
                        #print (flag)
                        #print(ear)
                        if flag >= frame_check:
                                cv2.putText(frame, "ASLEEP", (10, 30),
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)


                else:
                        flag = 0
                        #print(ear)
                        cv2.putText(frame, "AWAKE", (10, 30),
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):

                break
cv2.destroyAllWindows()
cap.release()
```