ILLINOIS INSTITUTE OF TECHNOLOGY

CS 584: MACHINE LEARNING

Final Project Report

MUSIC RECOMMENDATION SYSTEM USING EMOTION RECOGNITION

Author:
Dikshitha Varman – A20521565
Jayasurya R – A20513480
Kiran Gopi – A20520561

Supervisor:
Professor Yan Yan

1 INTRODUCTION:

1.1 ABSTRACT

Managing a vast music collection can pose a significant challenge for listeners, who must expend considerable effort in manually creating and organizing playlists containing hundreds of songs. The task of keeping track of unused songs can also prove daunting, resulting in wasted device memory. Moreover, selecting songs based on one's current mood and preference can be a cumbersome process that needs to be done repeatedly. Additionally, changing preferences and musical tastes can make it challenging to reorganize and play music effectively. To tackle these challenges, we have developed a machine learning approach that leverages facial scanning and feature tracking to detect the user's mood and generate a personalized playlist tailored to their emotional state.

1.2 LITERATURE SURVEY

"Facial Landmark Detection by Deep Multi-task Learning" by Z. Zhang is a paper published in CVPR 2014 that proposes a deep multi-task learning approach for accurate and efficient facial landmark detection. The method uses a single convolutional neural network (CNN) that simultaneously learns to detect 49 specific landmarks on a face, such as the corners of the eyes and the nose tip. The CNN architecture is designed to optimize multiple tasks, including landmark detection, pose estimation, and facial attribute prediction, by sharing lower-level feature representations. The authors evaluated their approach on two challenging datasets, AFLW and COFW, achieving state-of-the-art performance in both datasets. Specifically, the proposed method achieved a mean error of 3.28 pixels on the AFLW dataset and a mean error of 3.48 pixels on the COFW dataset. The results show that the proposed deep multi-task learning approach can significantly improve facial landmark detection accuracy while being computationally efficient. The paper concludes that the proposed approach can be used in various computer vision applications, such as facial recognition, emotion recognition, and facial expression analysis.

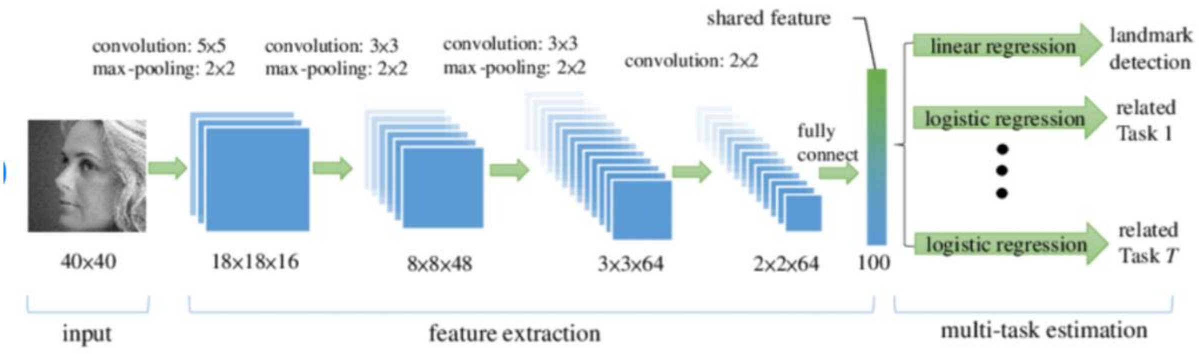


Fig1: Structure specification for TCDCN.

"Emotion Recognition using Facial Landmarks, Python, DLib, and OpenCV" by A. Jain is a paper published in ICCV Workshops 2017 that proposes an approach for emotion recognition using facial landmarks extracted from the DLib and OpenCV libraries. The method uses a pre-trained facial landmark detector from the DLib library to extract 68 points on a face, representing various facial features. The extracted landmarks are then used as input to a support vector machine (SVM) classifier that maps them to one of the seven basic emotions (anger, disgust, fear, happiness, sadness, surprise, and neutral). The authors evaluated their approach on the Oulu-CASIA dataset, achieving an overall accuracy of 60.34%, which is comparable to the state-of-the-art approaches at that time. The proposed approach is computationally efficient and can run in real-time, making it suitable for real-world applications. The paper concludes that combining facial landmark detection with machine learning algorithms can improve the accuracy of emotion recognition systems.

"Face recognition-based music recommendation system using user emotion and context awareness" by Y. Park et al. (2018) is a paper that proposes a music recommendation system that uses user emotion and context awareness based on face recognition technology. The method involves three main steps: face detection and recognition, emotion detection, and context awareness. The face detection and recognition module uses OpenCV's Haar Cascade classifier and Eigenface algorithm to detect and recognize faces. The emotion detection module uses the Microsoft Emotion API to detect user emotions from facial expressions. Finally, the context awareness module considers the time of day, location, and weather conditions to provide more personalized music recommendations. The authors evaluated their approach by conducting a user study involving 30 participants. The results showed that the proposed system was effective in recommending music based on user emotions and context awareness. The paper concludes that the proposed approach can improve the user experience of music recommendation systems by providing more personalized and relevant recommendations.

"Affective music recommendation using audio and lyrics features with long short-term memory neural network" by K. Choi et al. (2017) is a paper that proposes a music recommendation system that uses audio and lyrics features with long short-term memory (LSTM) neural network for affective music recommendation. The method uses audio features, such as spectral centroid and zero-crossing rate, and lyrics features, such as sentiment score and term frequency-inverse document frequency (TF-IDF), as input to an LSTM-based neural network. The LSTM model is designed to capture the temporal dependencies between the audio and lyrics features and predict the valence and arousal levels of a music track, which are indicators of the emotional content of the music. The authors evaluated their approach on the Million Song Dataset and the MusiXmatch dataset, achieving state-of-the-art performance in both datasets. Specifically, the proposed approach achieved a Pearson correlation coefficient of 0.414 for valence and 0.308 for arousal on the Million Song Dataset, and a correlation coefficient of 0.452 for valence and 0.358 for arousal on the MusiXmatch dataset. The results show that the proposed approach can effectively predict the emotional content of a music track and provide more personalized and relevant recommendations. The paper concludes that the proposed approach has the potential to be applied in various music-related applications, such as music therapy and music education.

2. PROBLEM DESCRIPTION:

The proposed system makes use of facial landmark extraction from the user's facial expressions, followed by classification to ascertain the user's sentiment. Once the user's emotion has been identified, songs that are appropriate for that emotion are displayed to them, possibly lowering their stress levels and facilitating more effective music selection. The three modules that make up the proposed architecture are Emotion extraction, Audio extraction, and Emotion-Audio extraction. The classifier should be supplied an image with a minimum quality of 320p that was taken in a well-lit location to achieve accurate findings. Users can save time and effort by not having to search or explore for songs thanks to the system.

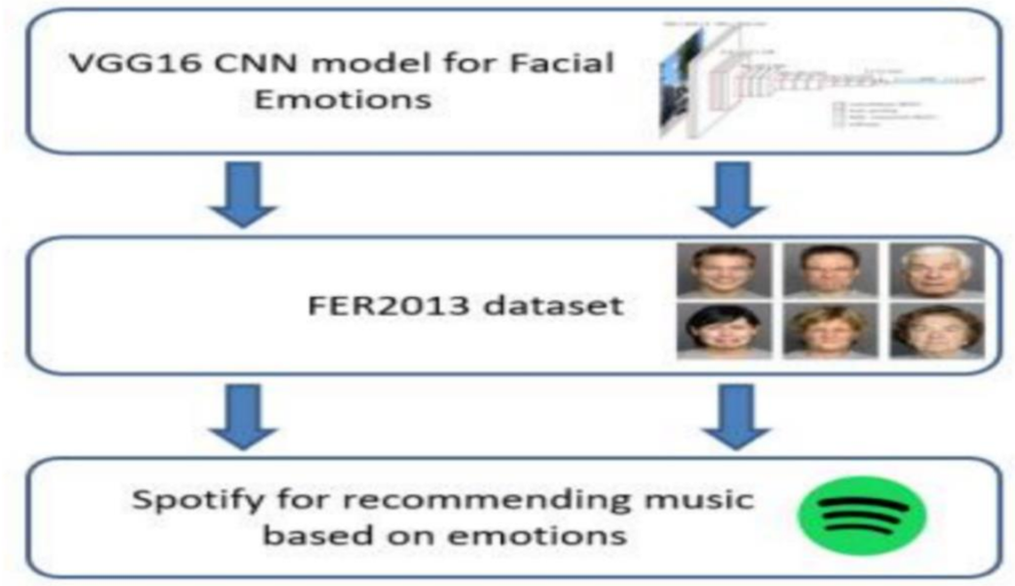

VGG16 CNN model for Facial Emotions
FER2013 dataset
Spotify for recommending music based on emotions

Fig 2.1: Proposed System

2.1 PROPOSED MODEL:

The primary webpage for this project includes a start video stream button on it that users must click in order to record their video broadcast. When a person clicks the capture button, an image is taken and transmitted to the server, where the VGG 16 model processes it and a forecast is created, allowing the user to determine their mood. The emotion labels and the matching prediction are matched, and the emotion is presented on the webpage. There are currently 7 emotion labels: surprised, disgusted, angry, happy, sad, neutral, and distasteful. The next step is to play songs based on the mood after the emotion has been identified. Our website has been integrated with the Spotify API. The client side then requests songs from the Spotify API, analyzes the songs' audio properties, and plays the selected songs to the user in attendance in accordance with the emotions discovered. Here, the user can either provide the playlist, and the songs from that list will be played based on the emotion identified, or a default playlist is applied. Additionally, the user could Play and Pause whenever necessary. Five components make up the system architecture: client, user, server, VGG16 model, and Spotify API.

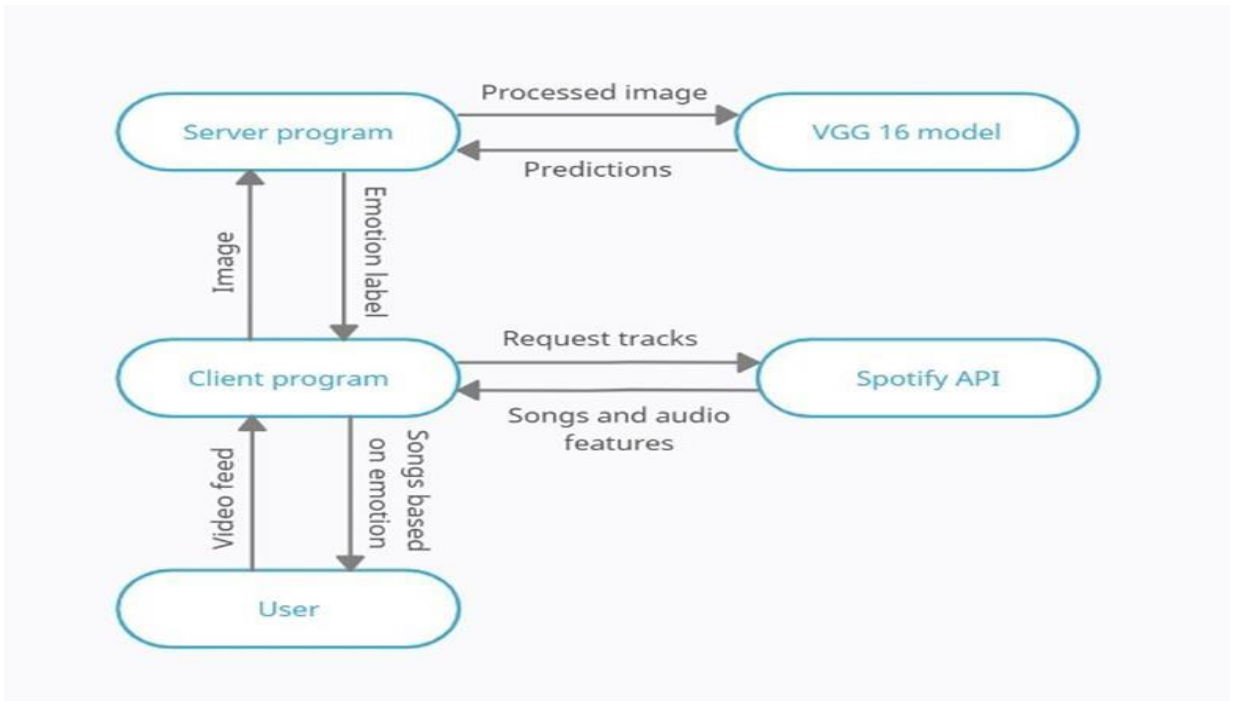
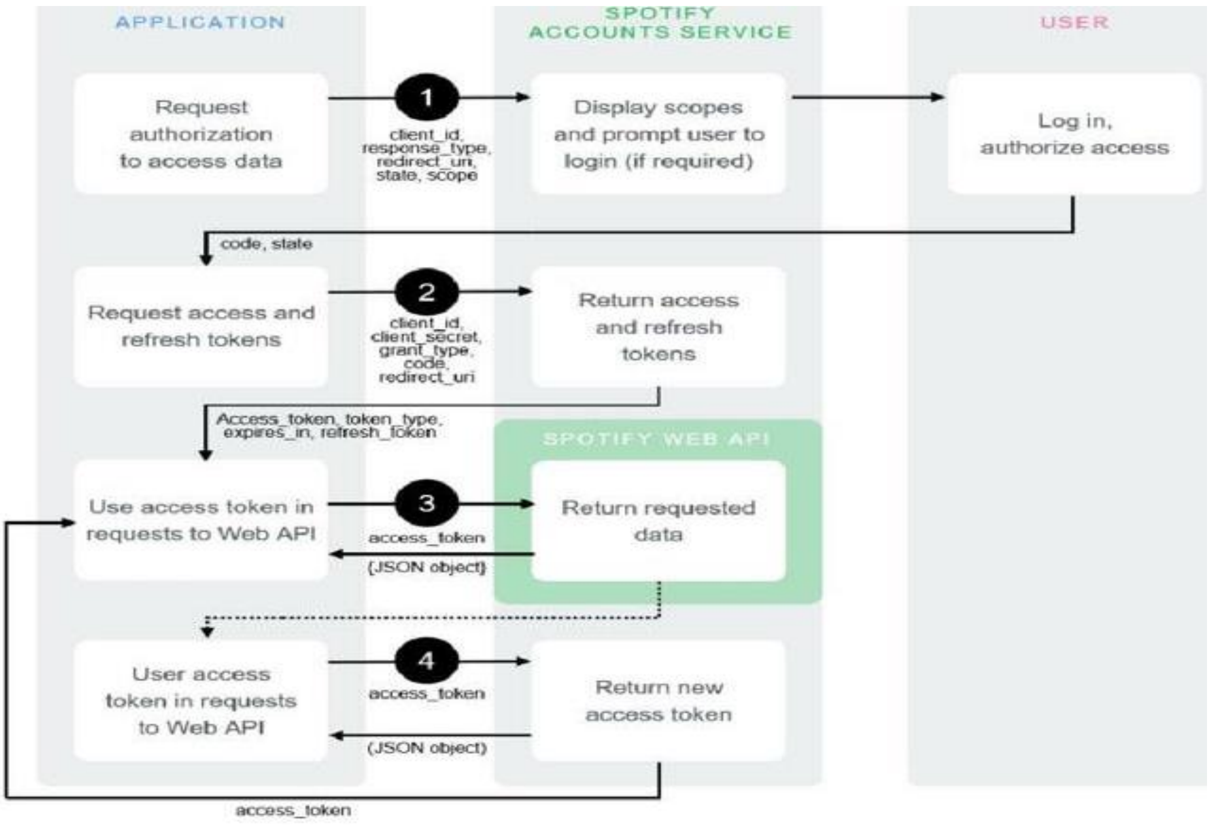


Fig -2.2: System Architecture of Music Recommendation System Using Emotion Recognition

A test application must first be created on Spotify's developer site, https://developer.spotify.com, in order to obtain data from the account service. This application must be used for all requests. The access token, which is obtained by providing the client_id, client_secret_id, grant types, and scopes, must be given along with every request. Using this, you can download songs from a playlist, access audio features, learn more about the songs, and more.


APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

2.2 SYSTEM IMPLEMENTATION:

a. **EMOTION DETECTION:** Using a pre-trained VGG16 model and a HaarCascade classifier from OpenCV, the system has an "emotion function" that can identify various emotions. It processes an image that is provided as input before applying the classifier. The model performs a prediction and outputs an emotion label if a face is discovered.

b. **SPOTIFY API:** The system accesses data about tracks, gadgets, and audio features using the Spotify API. To get an access token for authentication, it employs the two procedures "requestAuthorization()" and "callAuthorizationApi()". Sending requests to the Spotify accounts service for different kinds of information is done using the "callApi()" function.

c. **FLASK FRAMEWORK:** The Flask framework is used to build the system's server. "Home" and "Emotion" are its two pathways. While the "emotion" route receives an image and returns an emotion label, the "home" route returns the application's webpage. The "emotion function" is used by the "emotion" route to identify emotions.

d. **VGG16 MODEL:** To detect emotions, the VGG16 model was trained on 478 photos over the course of more than 400 epochs. Throughout training, various hyperparameters were adjusted. On the train set, the model had an accuracy of about 98%, and on the test set, it had an accuracy of about 70%.

e. **IMAGE PROCESSING:** The "emotion function" pre-processes the input image before the HaarCascade classifier is used. A "nil" label is returned if no faces are detected. The "emotion function" feeds the most recent face discovered to the VGG16 model for emotion detection if a face is detected. An emotion label is produced after analysis of the predictions.

3. THEORETICAL DERIVATION OF ALGORITHM(BONUS):

Convolutional neural network (CNN) model emo_model, with 16 layers, is defined by our team's code. With certain alterations, the model is based on the VGG16 architecture. The approach aims to categorize face expressions into 7 groups. Convolutional layers with various filter sizes are the foundation of the model architecture, which is then followed by batch normalization and max-pooling layers. L2 regularization with a 0.01 coefficient is used to regularize the convolutional layers. The final layer of the model, which has seven output units with a SoftMax activation function, has three fully connected layers. The stochastic gradient descent (SGD) optimizer is used to train the model, using a learning rate of 0.0001, momentum of 0.85, and Nesterov momentum. Accuracy is employed as the evaluation metric, while categorical cross-entropy is the applied loss function. The ImageDataGenerator class of TensorFlow is used to create the training data, and it employs data augmentation methods like zooming, shifting, and flipping to expand the training dataset. With a batch size of 113 and 400 epochs, the model is trained using the fit () method, with early termination depending on validation accuracy. Theoretically, this code develops a CNN model that is based on VGG16 and is capable of accurately identifying facial expressions. The generalization performance of the model is enhanced using batch normalization and L2 regularization. The strategies used for data augmentation during training assist in reducing overfitting and enhancing the model's capacity to generalize to new data. The SGD optimizer with momentum and Nesterov acceleration helps to speed up the convergence of the model during training. Convolutional layers with batch normalization, layers with maximum pooling, and fully linked (dense) layers make up the majority of the model. The stochastic gradient descent optimizer (tf.keras.optimizers.SGD) is then used to assemble the model with a learning rate of 0.0001, momentum of 0.85, and Nesterov momentum enabled. The accuracy metric is used to assess the model's performance, and the categorical cross-entropy loss (tf.keras.losses.CategoricalCrossentropy) is the chosen loss function.

3.1 TIME COMPLEXITY:

We must examine the number of operations carried out by the algorithm as a function of the volume of the input data in order to determine the temporal complexity of the given code. Images are used as the input data in this case and are trained using a deep convolutional neural network model. The number of layers, the size of each layer, the number of parameters, the batch size, the optimizer employed, and the number of epochs all affect how time-consuming it is to train a deep neural network. The model has 13 convolutional layers in the code provided, followed by 3 fully connected layers and a SoftMax activation function. Each convolutional layer increasingly uses more filters, going from 64 to 512. Each filter is 3x3 in size, with "same" padding applied. Each layer is subjected to batch normalization and L2 regularization in order to enhance performance and decrease overfitting. A stochastic gradient descent (SGD) optimizer with Nesterov momentum and a learning rate of 0.0001 is used to train the model.

The temporal complexity of training the model can be calculated as O (N * E * P * L2), where N is the total number of training examples, E is the total number of epochs, P is the total number of model parameters, and L is the largest size that each layer can be. The number of steps per epoch in the provided code is equal to the number of training instances divided by the batch size, which in this case is 113. The number of images in the training set, x_train.shape[0], serves as the value of N for the training set. The value of E is 400, which represents the quantity of training epochs. The number of parameters in each layer of the model can be added together to determine the value of P. The greatest size of any feature map in the model, which in this case is 28x28, can be used to estimate the value of L. As a result, it is possible to estimate the time complexity of training the model as O (N * E * P * L2) = O (28709 * 400 * 13491947 * 784) = O (2.28 * 1020), which is a very huge number. This is simply a rough estimate, and the real time complexity may vary based on several variables, including the hardware used for training, the model's specific configuration, and the training optimization methods.

4. RESULTS:

With 478 images in each batch, the VGG16 model has been trained for more than 400 epochs. Different hyperparameters are tuned during training. For training, the Google Collab platform is employed. Below is a plot of accuracy graphs that were recorded during training.

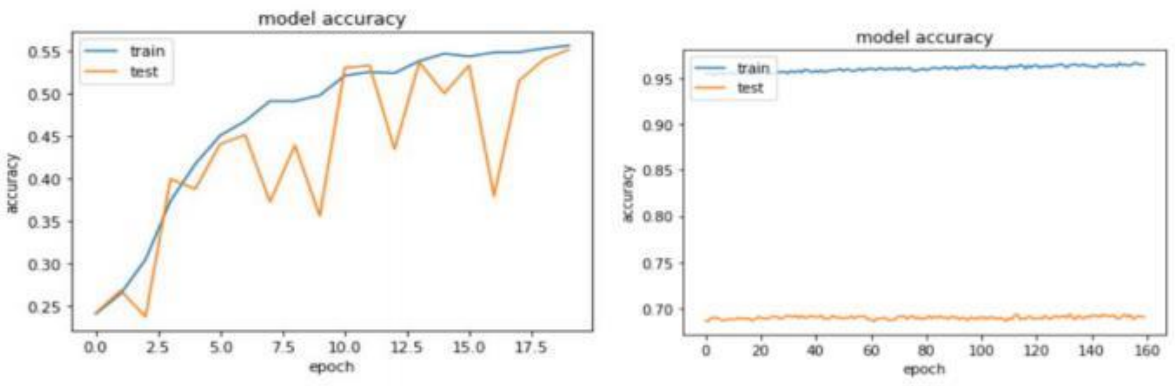
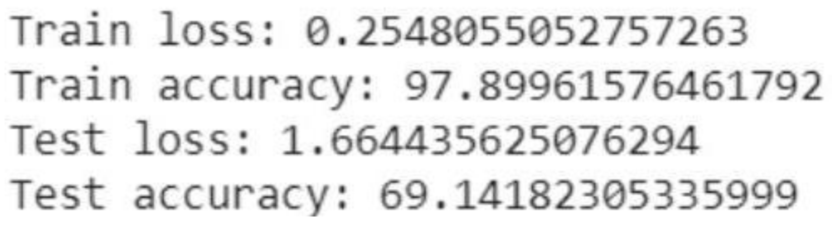


Fig 4.1: The left Snapshot is Epoch 1 – 20, where learning_rate = 0.01, momentum = 0.95, epochs = 20 and the right snapshot is Epoch 160 – 240, where learning_rate = 0.00001, momentum = 0.95, epochs = 80.

Our VGG16 model achieved an accuracy of around 98% in Train set and around 70% in Test set.



```
Train loss: 0.2548055052757263
Train accuracy: 97.89961576461792
Test loss: 1.664435625076294
Test accuracy: 69.14182305335999
```

Fig 4.2: Final test & train accuracy

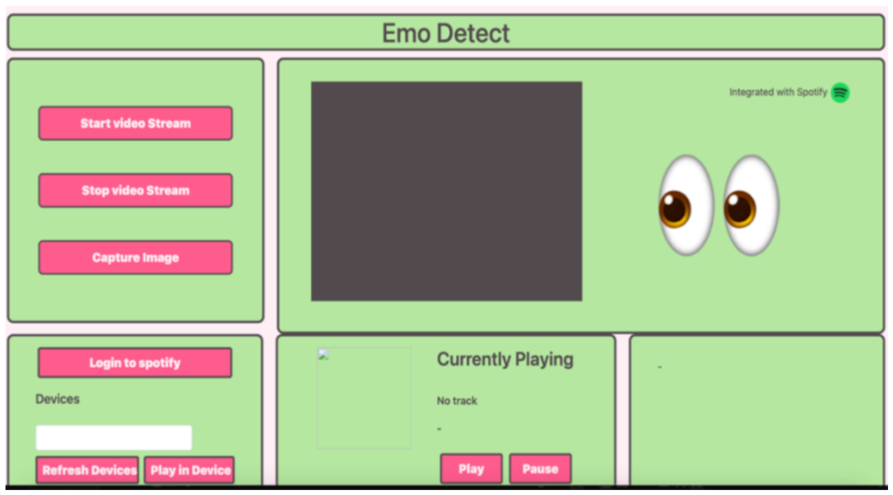


Fig 4.3 Homepage of our application

Below are some screenshots taken from the application where the user must start the video stream and capture the image. The emotion is detected, and songs will be played from the Spotify playlist.

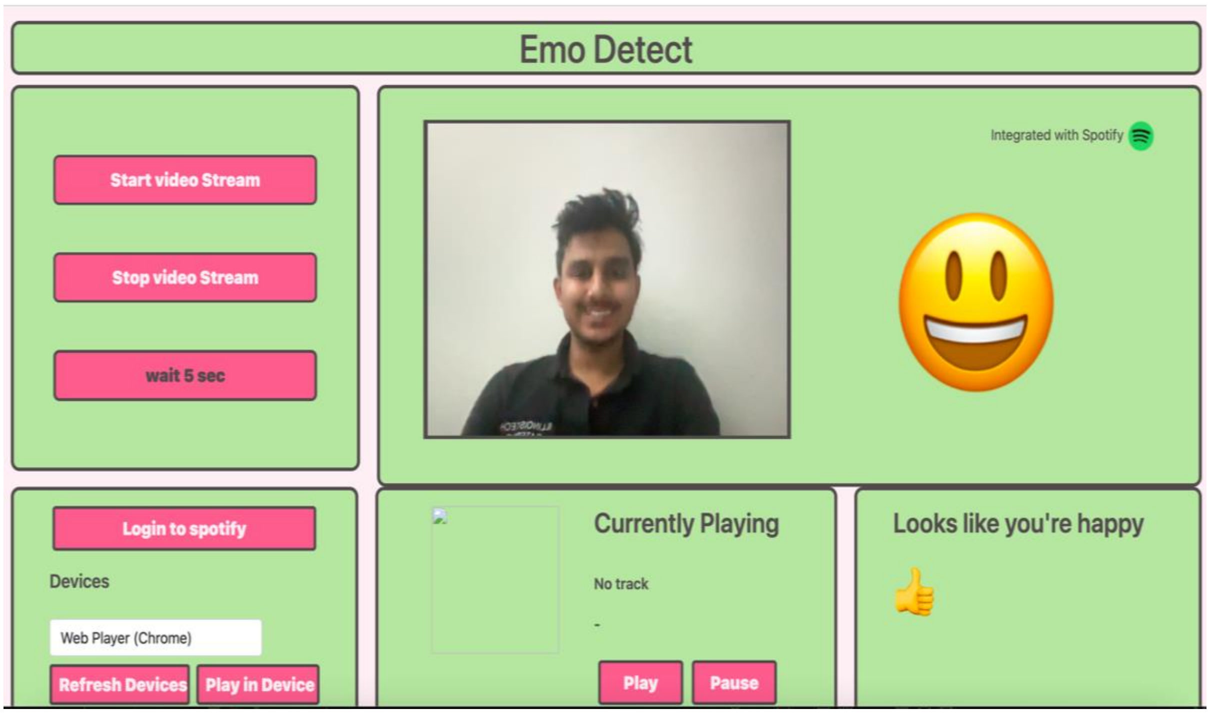
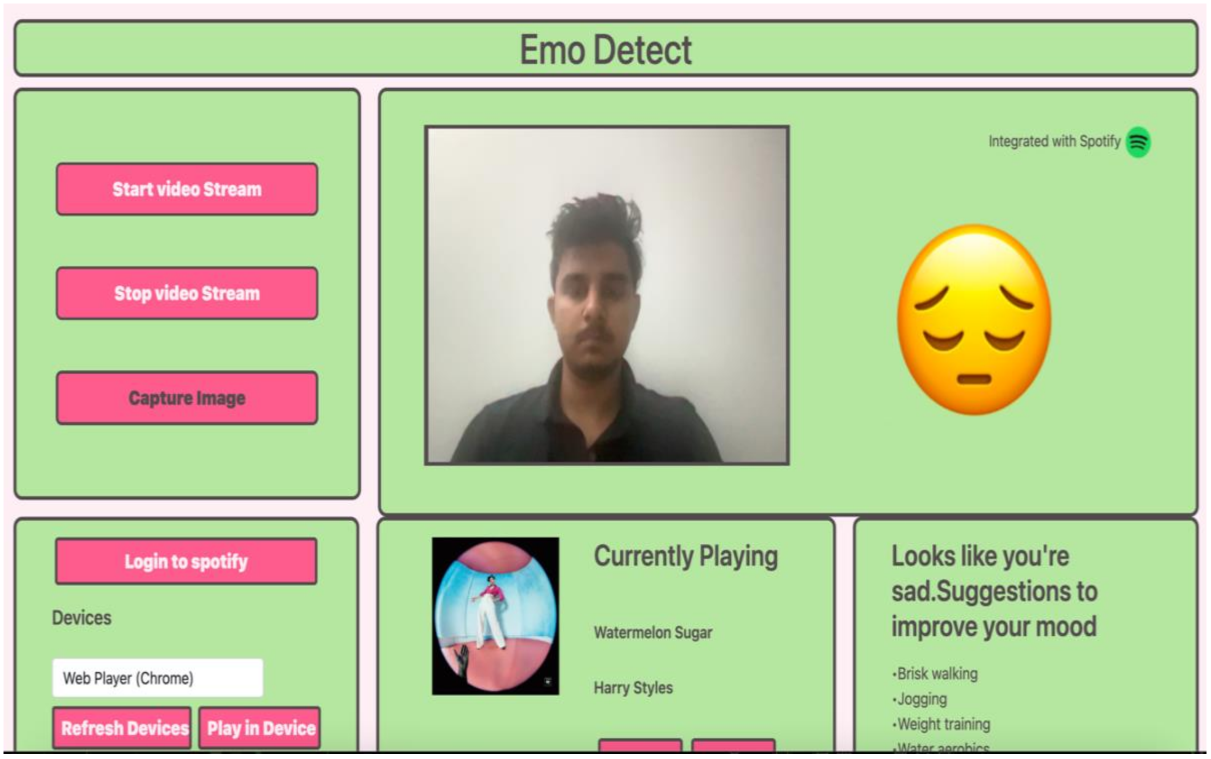


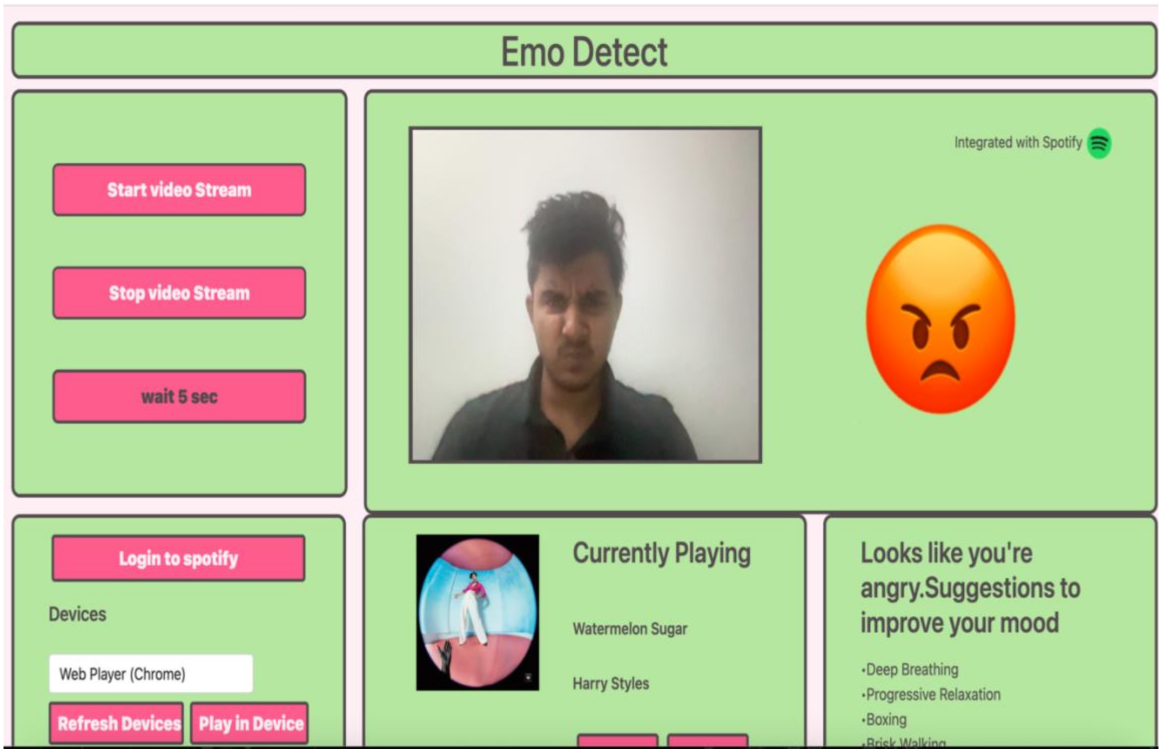
Fig 4.4 Emotion Detected: Happy



Looks like you're sad.Suggestions to improve your mood

- Brisk walking
- Jogging
- Weight training
- Water aerobics

Fig 4.5 Emotion Detected: Sad



Looks like you're angry.Suggestions to improve your mood

- Deep Breathing
- Progressive Relaxation
- Boxing
- Brisk Walking

Fig 4.6 Emotion Detected: Angry

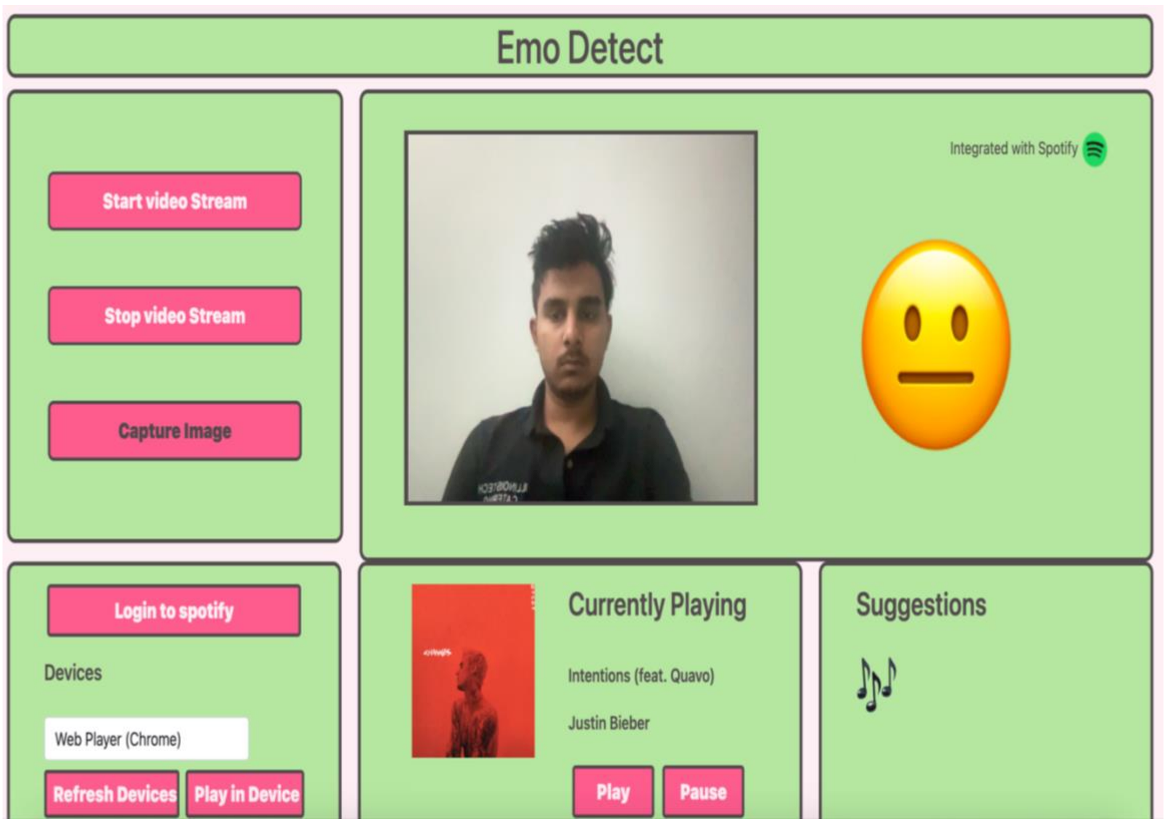


Fig 4.7 Emotion Detected: Neutral

5. CONCLUSION AND FUTURE SYSTEM IMPLEMENTATION:

The software's goal is to give operating system users a more user-friendly, stable, additional hardware-free emotional music system. The program for music driven by emotion and emotional behavior will be helpful to anyone looking for this type of music. The system's overall accuracy and efficiency could be improved by reducing the amount of time needed to search for music and, as a result, for unneeded computations. The application employs technology to strengthen the interaction of the system with the user in various ways, meeting the fundamental needs of music listeners without bothering them as do other programs.

5.1 TEAM CONTRIBUTION:

Our team worked collaboratively to build a comprehensive system that detects the user's emotions and assigns each song to its dedicated emotion playlist on Spotify.
Dikshitha implemented the emotion function using a pre-trained VGG16 model and HaarCascade classifier from OpenCV.
Jaysurya integrated the Spotify API to access information about tracks, devices, and audio features.
Kiran built the server using the Flask framework, designed the webpage, developed the routes, and ensured the smooth functioning of the system.
Throughout the project, the team members actively communicated and supported each other, ensuring everyone was on the same page and working towards the same goals. For instance, Dikshitha and Jaysurya worked together to fine-tune the VGG16 model and test its accuracy, while Kiran provided feedback and support throughout the development process. Similarly, Jaysurya and Kiran collaborated to integrate the Spotify API into the system, with Dikshitha providing valuable insights into how the emotion detection module could be integrated into the process. As a result of teamwork, our team was able to create a well-integrated system that leveraged the strengths of each member to achieve a common goal.In summary, the team members contributed equally to the project, supporting and collaborating with each other to build a robust system that accurately detects the user's emotions and assigns each song to its dedicated emotion playlist on Spotify. During the tenure of the project, we learned the following things:

a. The accuracy of the emotion detection module heavily depends on the quality of the input data. As such, it is important to preprocess the data before feeding it into the model to ensure that it is in the correct format, and all irrelevant features are removed. As a result, the model's accuracy is enhanced.
b. Hyperparameters can have a significant impact on the performance of machine learning models. Therefore, it is important to conduct thorough experimentation to identify the best hyperparameters for the model.
c. Integrating APIs was challenging for our team, especially when dealing with authentication and authorization. In this project, we encountered some challenges in obtaining the access token for authentication purposes. As such, it is important to carefully read the API documentation and follow the instructions to ensure the integration is done correctly.
d. Developing a web application requires careful planning and organization. Kiran had to design the webpage, develop the routes, and ensure the smooth functioning of the system. It is important to have a clear understanding of the user requirements and the expected behavior of the system to build a well-organized and user-friendly web application.

e. The success of our project heavily relied on the ability of our team communication and how we collaborated effectively. Regular feedback sessions, and brainstorming sessions were essential in keeping everyone aligned and moving forward toward a common goal.

5.2 FUTURE SYSTEM IMPLEMENTATION:

- Multiple face detection must be handled.
- Can be used to determine the mood of physically challenged & mentally challenged people.
- Music classifier models can be developed.

6. REFERANCES:

- Zhang, Z. (2014). Facial Landmark Detection by Deep Multi-task Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, 2175-2182.
- Jain, A. (2018). Emotion Recognition using Facial Landmarks, Python, DLib, and OpenCV. Available online: https://towardsdatascience.com/emotion-recognition-using-facial-landmarks-python-dlib-opencv-7750f73dc9d5
- Park, Y., Lee, J., & Lee, J. (2018). Face recognition-based music recommendation system using user emotion and context awareness. Sensors, 18(5), 1392. DOI: 10.3390/s18051392.
- Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017). Affective music recommendation using audio and lyrics features with long short-term memory neural network. In Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR), 2017, 109-116.

PROJECT GITHUB LINK: https://github.com/DikshithaVarmanA/CS-584-Machine-Learning-Final-Project-Spring-2023.git