Engagement Recognition Using Video-Based Expression Tracking

Akshaya Srinivasan, Sonal Sharma, Manisha Paliwal, and Praphul Kenkere Omkarmurthy

Department of Applied Data Science, San Jose State University

Data 270: Data Analytics Process

Dr. Eduardo Chan

May 17, 2022

Abstract

Recognizing a user's engagement with the system can change how the computer reacts to the viewer in such situations. This will not only increase user experience with the computer but would also allow for better human-computer connection. The scope of this project is to build machine learning and deep learning classification models to identify the intensity and type of engagement modes of humans from the videos during virtual interactions. In existing work, deep neural networks outperform traditional machine learning models in recognizing emotions. Traditionally the distance metrics used in KNN give equal importance to all features. This project presents an in-depth comparison between the performance of SVM, KNN, LSTM and EmotioNet algorithms in terms of effective prediction for video inputs and giving accurate intensities for each emotion for each video. Among the Machine Learning models, KNN with weights of SVM is better in terms of accuracy with KNN at 69 percent and SVM at 45 percent. With respect to deep learning models, multilayer Bi-LSTM proves to be far better than EmotionNet in terms of prediction accuracy with Bi-LSTM at 90 percent and EmotionNet at 58 percent. Overall, Bi-LSTM is the best model with the highest accuracy of 90 percent.

1. Introduction

1.1. Project Background and Execute Summary

Emotions are used by humans to socialize. It describes a person's engagement state as it pertains to his objectives or the physical efforts, he must make to complete activities. As a result, engagement recognition is beneficial in a wide range of applications, like e-learning, virtual conferences, cybersecurity, robotics, psychological investigations, and virtual reality applications (Mehta et al., 2018).

Empowering computer systems to identify and interpret expressions from facial gestures instantaneously is a challenging research area. The coronavirus pandemic has had a significant impact on how we live our daily lives to interact professionally, personally, socially, and educationally. Virtual reality can alleviate many of the issues posed by the pandemic, encouraging increased adoption. However, the capacity of applications to effectively address virtual engagement experiences is still limited due to low overall usage and inadequate software (Ball et al., 2021).

We present an artificial intelligence-based Engagement Recognition framework in this study. Given the fact that other datasets are available, we chose the DAiSEE dataset for affective state recognition as it captures user expressions in the wild. We will explain different ways of feature extraction from videos in real-time and provide a comparative study of different classification approaches. We plan to retrieve configural features from a series of images by combining tools like OpenCV, OpenFace, Gabor Filters, FaceNet, and 68 facial landmarks dlib, Custom CNN using Kera's sequential method. By doing this we will derive the feature vectors that define the shape and shading changes of action units in the face. This work aims to perform a comparative analysis of classification using traditional Machine Learning approaches such as

SVM and kNN Algorithm, as well as deep learning algorithms such as LSTM and EmotioNet, to recognize the emotion as well as the respective intensities of those emotions.

This will serve as a starting point for developing an adequate model for applications to enhance human-machine interaction. Identifying consumer experience can be instrumental in a variety of applications, such as healthcare, driverless cars, marketing, and e-learning. For example, conferencing applications such as Zoom, Microsoft Teams, and others, that capture live video can interpret the nonverbal responses of conference participants and give a live conference-like experience. It will also help virtual reality applications to identify their nonengaging/weak areas and get productive feedback to improvise the products.

1.2. Project Requirements

There is a myriad of datasets available for video-based engagement recognition, amongst which we have chosen the DAiSEE dataset because it captures users' expressions in the wild. Most of the other datasets had a mediocre performance in real-world scenarios. The requirement is to have videos of varying quality, illumination, and backgrounds to have well-rounded training data for the model to train on. So, it was necessary to get a dataset that best portrays real-world conditions. Human emotions can be classified as sadness, happiness, fear, anger, surprise, and disgust. If not these, the emotion is categorized as neutral.

DAiSEE dataset has 9068 video snippets, and it is a multi-label video classification. The authors collected these samples from 112 users and classified them into four-level intensity scales for the purpose of determining user involvement such as frustrated, engaged, bored, and bewildered. The dataset has also had labels representing the intensity of engagement into 4 levels. This dataset contains 2,723,882 videos extracted against 9068 clips and 25 hours of video, and this is large enough to train some of the best deep learning models. The age group of these

participants varies from 18-to 30 from the Asian race and they are recorded in various locations (Gupta et al., 2018). The project is based on supervised learning techniques; therefore, it is required that each video in the dataset is labelled to represent the intensity of engagement into 4 levels.

Users referring to this paperwork need to be familiar with the mathematics behind image processing, vectorizing images to extract features, preprocessing methods such as photometric normalization, histogram manipulation dimensionality reduction. Conceptual and applied knowledge of machine learning and deep learning algorithms including but not restricted to SVM, LSTM and CNN are crucial. Also, familiarity with various model evaluation metrics such as confusion matrix, accuracy, recall, area under the curve, etc. would be required.

1.3. Project Deliverables

The project performs a detailed comparative study of different feature extraction methods for video and image data, and different classification algorithms. We are using agile methodology to perform project management. Each sprint consists of two weeks and deliverables will be provided in each sprint. Project execution plan prototype using a Gantt or PERT chart and an extensive work breakdown structure will be delivered. Code components like python notebook, dataset link, result set are part of deliverables. A complete research report on the project comprising details of the approach, different feature extraction methods for video and image data, and different classification algorithms will be delivered. The report also includes a study on performance of different models using validation and evaluation metrics.

1.4. Technology and Solution Survey

Existing research work provides a variety of solutions to each of the above steps involved in our problem statement. Patel suggests using the Python framework called OpenCV to convert

videos to images. With this library, we can experiment with custom frames per second(fps) values to find the most optimal one for the video. Ideally, 0.5 is chosen as the fps value, which will generate 2 images for every second of the video. For our dataset, with each video being 10 seconds long, this will translate to 20 images per video (Patel, 2019).

The next two steps of feature extraction and model generation go hand in hand based on the machine learning algorithm used and the type of features the model accepts. In fact, some machine learning algorithms are coded in libraries that have the capability to automatically extract the features without having to write explicit lines of code. One way of going about extracting features is to identify the Regions of Interest (ROI) in a person's face and work with them to identify the emotion portrayed. In a research work by Lekdioui and team, the authors present a method to detect facial expressions by decomposing the facial space into regions of interest. They have used the IntraFace (IF) framework to decompose the facial space and effectively detect different components of the face, irrespective of the shape and size of the face. The framework uses the Supervised Descent Method to decompose the face and outputs 49 markers of essential regions such as eyes, nose, eyebrows, lips, etc. Once these ROIs are identified, the authors perform transformations such as scaling and partitioning them. The key feature descriptors are extracted and merged for further analysis. The authors then develop an SVM based classifier that can categorize the image based on these extracted features into one of the 7 emotional states. On comparison, the recognition rate of the developed model was a stunning 96.7% against the 90.08%, 94.1% and 93.7% detection rates from the existing deep learning frameworks from Happy and Routray, Ghimire and Liu respectively. The authors have also performed cross database evaluation against the popular standard CK, FEED and KDEF. On

evaluation against these datasets, the authors achieved accuracy of 96.06%, 92.03% and 93.34% respectively in emotion detection (Lekdioui et al., 2017).

In another research work by Rajesh and Naveenkumar, the authors have used Principal Component Analysis (PCA) to reduce the dimensionality of images before modeling. This reduces the number of features per image, model complexity and processing time. They have also used Linear Discriminant Analysis (LDA) is used to solve a variety of recognition issues by calculating a set of characteristic features that standardize the different classes of picture data for classification. They have used SVM as well for creating the classifier and categorizing emotions. While the recognition accuracy of the model was not disclosed by the authors, the focus of the paper was on reducing the processing time at each stage. The time estimate for detecting the face, extracting the features, classifying using the SVM model and emotion detection are 0.084, 0.921, 0.195 and 0.199 seconds respectively using the proposed framework (Rajesh. & Naveenkumar, 2017).

Moving on to neural network approaches for feature extraction and classification of emotions, Jaiswal and team employed a CNN based model for engagement detection. In their work, the authors have used two datasets, Japanese Female Face Expression (JAFFE) and Facial Expression Recognition Challenge (FERC-2013) as inputs for emotion recognition to train on images with varying illumination, consistency, and cleanliness. The authors have built a CNN based model that contains two sub models, whose outputs are reduced as vectors, combined into a single matrix, and sent to a final layer for recognition and classification. This model has 4 layers containing 64 filters, all with a dimension of [3*3], a normalization layer, a max pooling layer, and another convolutional layer. Following that, they integrated comparable models to an activation layer to categorize images into one of the seven emotions. The model produced

70.14% accuracy with the JAFFE dataset and 67.02% accuracy with the FERC dataset (Jaiswal et al., 2020).

Burkert and team have presented a CNN based emotion detection model that does not need explicit feature extraction. The core of their framework lies in using Parallel Feature Extraction blocks called FeatEx, in which two blocks, each consisting of 5 layers of convolutional and pooling filters are combined to extract key features from any input image automatically. The extracted features are then sent to a fully connected layer for classification. The researchers have achieved an accuracy of 99.6% for Extended Cohn-Kanade and 98.63% for MMI datasets (Burkert et al., 2016).

Based on the solutions and methods from the research work discussed in this section, we can see that with both conventional and deep learning-based classification approaches, prominent levels of recognition rates can be achieved. The key to achieving elevated levels of accuracy relies on the parameter tuning of the models, along with choosing the right method of feature extraction for that data. For instance, although the first two papers discussed in this section employ SVM based classifiers, the chosen feature extraction methods helped them achieve their respective goals of better accuracy and lower processing time. Similarly, although the last two papers use CNN based model for classification, using deep learning techniques for automatic feature extraction has proven to give a better accuracy as compared to manual vector-based extraction.

The solution to our proposed project involves a series of transformations and processing on the dataset. Each of our data records is a video of 10 seconds length and is labeled with emotion and its intensity. The first step in our solution would be to convert these videos into image frames with optimal frames per second value. Once we have the images, feature extraction

needs to be performed to identify key aspects of the image that would be instrumental for classification. Finally, conventional machine learning and deep learning models would be developed and trained using the extracted features to classify human emotions. The trained model will then be used to classify new test data points.

1.5. Literature Survey of Existing Research

Analyzing an individual's affective state using machine learning methods and artificial intelligence techniques has been studied for years. This paper utilizes user affective states of boredom, perplexity, engagement, and displeasure on a scale from very low to very high for each state as provided by DAiSEE. With the increase in the use of virtual platforms for daily activities, this domain has gained popularity in research. We have referred to various previous research works and their computational implementations, suggestions, and conclusions to gain insights into the subject (Gupta et al., 2018).

During the early years of research in this domain, Hernandez examined the possibility of using visual data to accurately estimate the level of involvement of TV viewers. The custom dataset (limited for use of their research) is created by using an RGB camera and manually labeling the engagement levels of TV viewers. They framed the task of detecting a TV viewer's involvement as a binary classification problem, applying face geometry characteristics with SVMs for classification. Head and facial gestures are used for the analysis. The difference between low and high involvement levels was identified with an accuracy of 79.3 % with the use of a particular feature while 82.54% when a subset of features was used. This method of classification proved better than the conventional approach of using head pose which accounts for only 62.37% accuracy. With the use of an appropriate combination of features the overall accuracy of the model achieved was 77.92%. A productive classification model was

implemented with just five facial points to minimize cost. Although more features could provide better insights, the research proved the point that automatic engagement is doable with simpler complexity (Hernandez et al.,2013).

Whitehill and the team aimed to study participation in learning settings automatically. They created a project-specific dataset labeled by coders. They use different methods to extract various features and classifiers. According to their research on evaluating user engagement, SVMs with Gabor features provided the best results, around 69%. However, the dataset used was collected under limited conditions and does not represent real-world user engagement identification (Whitehill et al., 2014).

In (Gupta et al., 2018), the authors introduce their dataset DAiSEE, which was collected from 112 users and classified as frustrated, engaged, bored, or bewildered on a four-level intensity scale for the purpose of determining user involvement. They have also integrated classification techniques to help forecast a user's level of engagement. Two Inception Net V3 models were developed, each of which was trained with a separate sequence of frames and was derived from CNN and pre- trained using ImageNet. For video categorization, a C3D model and a Long-Term Recurrent Convolutional Network were created. The authors attained a 73.09%, 57.45%, 51.07%, and 35.89% accuracy in identifying frustration, confusion, engagement, and boredom, respectively, based on the comparative analysis between the models (Gupta et al., 2018).

On the DAiSEE dataset, Xiaoyang and team suggested an engagement recognition approach utilizing Neural Turing Machine. For classification, the model looks at eye stare, head attitude, body posture, and facial activity. Open-faced and C3D networks receive input in the form of a series of continuous frames. The multi-modal features are then combined in a serial

fashion to form a complete feature. Gaze-AU-Pose (GAP) features were used to add body position attributes, and then video attributes were included to a deep network and a SoftMax layer to guarantee that the learnt numbers matched the probability distribution. A deep neural network learns all the weights of parameters by itself. Caused by the intensity of the small video attributes, the superior brief video attributes account for a greater share of the total video features, making it simpler to accurately gauge student engagement and improving the experiment's accuracy. In comparison to the traditional model, which has a valuation accuracy of 57.5% and a test accuracy of 58.1%, the model proposed in the study has a valuation accuracy of 60.2% and a test accuracy of 61.3%. (Xiaoyang Ma et al., 2021).

Dehghan and team brief Sighthound's fully automated age, gender, and emotion recognition system with low error rates. They depict how their algorithms outperform commercial and academic algorithms on different criteria using deep architectures and an in-house dataset that has over four million images collected from more than 40,000 people for face recognition, 600,000 images for age estimation, and four million labeled data for both gender and emotion recognition. This dataset is collected through a semi-supervised pipeline to reduce time and effort. Authors created separate networks for each task which allowed them to design faster and more robust models which take comparatively less time than the network that has been trained with all models together. The author's model has the least mean absolute error (MAE) of 5.76 compared to others such as Microsoft and Kairos having an MAE of 7.62 and 10.57 respectively. Similar results were observed for gender and emotion recognition (Dehghan et al., 2017).

In the research work by Li & Lam for facial emotion detection using deep learning, images are first preprocessed using photometric normalization. This preprocessing is a vital step

since it will remove any lighting related variance from the images. Variation in illumination condition can introduce major changes to any image, hence affecting the overall accuracy. Image is product of illumination and reflectance. Extraction of facial features was done by convolving images with Gabor filters and after that kernel PCA was applied. The emotions were then classified into six basic expressions like fear, anger etc. Confusion matrix was calculated to evaluate the performance of deep neural networks. As compared to SVM models, deep neural networks were much more efficient in recognizing facial expressions with recognition rate of 96.8% (Li & Lam, 2015).

A system for identifying emotions based on two kinds of data: speech and facial characteristics was proposed by Jiménez and the team. On an English voice-to-text task, a pre-trained xlsr-Wav2Vec 2.0 model was used to create a speech emotion recognizer. Action units were employed as features in the face emotion recognizer and were evaluated on two models: static and sequential. The results of two systems were integrated utilizing a late fusion strategy: a vocal emotion identifier and a facial expression recognizer. For Speech emotion recognizer three algorithms: SVC with 'RBF' kernel, k-NN and MLP were used. Action Units derived from the OpenFace utilizing static and sequential models were evaluated for facial emotion recognizers. SVC, k-NN, and MLP were used to assess static models by modifying their hyper parameters. A bi-LSTM of two layers with 50 neurons was built for the sequential model. For SER, using pre-trained and deeper models made significant differences in accuracy achieved. MLP with a single layer of 80 neurons had the highest accuracy of 58.93% in FER (Jiménez et al., 2021).

Mehta and team have performed a comparative study of face emotion identification classifiers and the intensity estimation of such emotions. Gabor filters and LBP for feature extraction were utilized in this comparative study. Algorithms based on Action Units were used

for measuring facial emotions and their intensities. LBP paired with SVM has the highest accuracy. The results also confirmed that the comparative study can be applied to real-time behavioral face expression and intensity recognition in the future for crime prediction systems and drowsy driver detection (Mehta et al., 2019).

The authors Quiroz and team present a computer vision algorithm in their paper, to extract Action units (AU) and AU intensities across multiple databases (CFEE, DISFA, CK+) and not just one. All the references taken in the paper by the authors can recognize only within databases that they are trained on because of the differences in filming conditions and variation in subject population. The authors used Five-fold-cross validation to test the accuracy of the algorithms. The proposed algorithm also runs in real-time (more than 30 images per second) which gives the flexibility to work with more videos and images (Quiroz et al., 2016).

Bosch & D'Mello have described how to use facial features to detect mind wandering. Mind wandering is drifting of attention from important tasks to unrelated things. Support Vector Machines achieved F1 score .478 and deep neural networks achieved F1 score .414. Detectors can be further integrated with intelligent interfaces to enhance engagement. Frequent mind wandering affects the overall productivity of a person. The goal of this paper was to develop methods through which mind wandering can be detected and to aid in various applications that will help in improving the overall performance in a particular task. SVM and DNN models were used for various features like motion, head pose, texture etc. L1 regularization was added to the first layer of DNN model to reduce the effect of ineffective features. TCFS was developed and trained on SVM for each feature and later the features were ranked based on accuracy of various feature models. If individual feature sets are considered, then SVM outperforms DNN models

though not significantly. The highest F1 score was of feature-level fusion models (Bosch & D'Mello, 2021).

Chen and the team researched remote PhotoPlethysmoGraphy (rPPG), which is a noncontact video-based method to measure Heart rate (HR) of a person in their paper. rPPG is quite a superior technique as it is a non-contact video-based method and hence there is no need for a patient to wear sensors which sometimes causes allergies, also the cameras used are low cost and convenient. rPPG measures the pulse rate by capturing the pixel intensity changes from the skin by monitoring the changes in blood volume. rPPG can also measure blood oxygen saturation, blood perfusion and blood pressure. rPPG are a convenient way of monitoring infants and elderly at home or hospitals. It can also be extended as a polygraph. Quasi-periodical pulse-induces subtle color variations that can be measured using a camera. Basic framework of rPPG : Camera is employed in well-lit room. Skin region of interest can then be tracked, and spatial means calculation is performed to derive pulse information which will finally generate the heart rate estimation. rPPG employing infrared cameras are perfect for sleep monitoring. Recent advancements in rPPG have paved the way to overcome challenges that arise from illumination variation and motion artifacts (Chen et al., 2019).

The difficulty of detecting student interest in prosocial games using engagement cues from various input modalities has been addressed by Psaltis and team in their paper. Modeling of real time data was done by considering different dimensions like behavioral and cognitive. Based on how students interact with the game their behavioral and cognitive features were extracted. Machine Learning approach based on artificial neural networks was used for automatic recognition of engagement and novel approach was used for annotation of the engagement data. In the case of video games, more engagement depicts that the user will be hooked to the screen

i.e he is more engaged. For the extraction of facial features Kinect SDK's face tracking engine was used and a neural network was trained to categorize emotions. The Bimodal late fusion scheme was also used to categorize emotions based on concatenation of facial features and body features. Classification accuracy was 85%, multimodal affective recognition is better than monomodal classifiers (Psaltis et al., 2018).

A generic framework for automatic demographic estimation is presented by Han and team in their paper. Demographic refers to a person's age, gender, and race from facial recognition. It has applications widely from forensics to social networking. It is a hard challenge to estimate age because people from the same demographic group might have significantly different facial traits because of many factors. The authors extract demographically important features from a face using a boosting technique and then employ a hierarchy of strategies that involve between-group classification and within-group regression. The authors perform a quality evaluation and data cleaning of the face to detect low-quality photos from which it is very hard to get demographic estimations. Finally, the authors make a random comparison of the algorithm with crowdsourced random demographic data (H. Han et al.,2015).

Zhang & Ling in their research work explore the importance of intelligent video surveillance in the computer vision field. Identifying human emotions correctly has been challenging in the past. Static and dynamic features were extracted using KTH behavioral dataset. Their findings reveal that human behavior based on deep learning-based identification algorithm has high recognition accuracy by properly extracting joint motion information. The dataset has various actions like clapping, boxing, jogging, etc., and the results show that out of all the actions, categorizing boxing action has the highest accuracy of 92.12% and the least accuracy was of jogging, 51.62 %. Boxing has the highest accuracy since it has more static

features and hence its spatial interpretation is much easier compared to jogging which has more dynamic characteristics (Zhang & Ling, 2020).

Lucey and the team develop a system that can detect if a patient is in pain or not. It has become extremely important to provide medical care at crucial times. Usually, the patients in the hospital suffer without getting proper medical care because neither do they report nor is there an observer to raise an alert. The authors have proposed facial action units to retract objective measures frame-by-frame. The authors have described the appearance model (AAM) which detects the frames from videos where the patient is in pain. This dataset contains people with shoulder injuries. But it cannot track the pain accurately because of the patient's limited mobility around the neck region because of the proximity to shoulder pain. It can also ensure that the AAM reports only if the pain intensity is greater than 10 or reports in seconds or minutes to increase the frequency of detection (Lucey et al., 2011).

We've seen that machine learning methods like SVM can be employed for classifying facial expressions. SVM can be used to detect emotions both in live video and images. Face localization and feature displacements extraction from a video was performed using feature tracker by Michel and Kaliouby in their research. Trained SVM model was then used to classify test feature displacements. Kernel functions are used to map input data. SVM has good accuracy even when a huge data set is not available. SVM are suitable for dynamic, interactive approach to emotion recognition. Selection of appropriate kernel function allows optimization of SVM classifier that helps in accurately classifying emotions. SVM was also used in classifying text and DNA analysis. Selecting appropriate kernel function improved the accuracy of model from 78% to 87.9%. If additional enhancement like head motion is considered, SVM would perform even better and would make it more suitable to recognize emotions (Michel & Kaliouby, 2003).

Based on the research papers discussed, we can conclude that previously Support Vector Machines, a traditional machine learning model can be employed to detect facial expression from live video or images. Automatic feature trackers can be used for face localization and feature displacements extraction from a video. SVM classifiers can be optimized by selecting appropriate kernel functions. In most cases, deep neural networks outperform SVM models in recognizing emotions in terms of accuracy. Using image preprocessing techniques such as photometric normalization has proved to be efficient for the researchers, as it helps in removing any sort of light related variance more effectively. Additionally, using a boosting technique and implementing between-group classification and within-group regression has proved to be instrumental in detecting features among people of different diversities. Extraction of facial features can be done by convolving images with Gabor filters. Before feeding images to multiple layers of deep neural network, Kernel PCA can be applied. The emotions can then be classified into the six basic expressions. Confusion matrix can then be calculated to evaluate the performance of the classifiers. Deep Neural Networks are better in generalizing new images. In most cases, using multiple layers of convolution filters along with pooling layers has helped the neural network model to identify emotions with increased accuracy as compared to lesser filter layers.

2. Data & Project Management Plan

2.1. Data Management Plan

There is a plethora of datasets available over the internet. We will be using DAiSEE dataset which has videos of different types. It has 9068 video snippets. It is very important to train the model in different light conditions. This dataset consists of 2,723,882 videos retrieved from 9068 clips. The larger the dataset, the better the deep learning and machine learning models will be trained. This dataset captures the emotions of 112 users. The videos were captured in different locations such as crowded places, dorms etc. The emotions were then categorized as happy, sad, bored etc.

Other datasets that are available for emotion detection is HBCU. It consists of 120 videos. The emotions categorized for this dataset then showed how engaged or disinterested the user was in the video. In-the-wild dataset has 120 videos of 34 users and the emotions categorized were disinterested, highly interested etc. SDMATH dataset has 20 videos of 20 users and the emotions were categorized as deictic gestures (Ma et al., 2022).

Data management is a crucial step in the implementation of any project. Data in its raw format might not be usable directly and hence most of the time there is a need to perform data wrangling. Data preprocessing and data normalization is performed to convert the raw data into format that can be fed to machine learning and deep learning models. Discrepancies in the data can lead to inconsistent results. There are numerous data storage tools available in the market like relational databases, NoSQL databases. Usage of SQL or NoSQL database depends entirely on the type of data that must be stored. When considering production environments, the primary focus is not just on data storage but also on the security and access to that data. In production environments database administrators are responsible for granting access to other non-admin

users. Not everyone is allowed to make changes to the production environment so usually non-admin users are just granted read access and permissions to fetch the data through some queries. DBA can grant write and delete access to admin users so that only admin users can make the changes, so my granting appropriate permissions to different user's data security is ensured on production environments. Further data security can be ensured by SSH tunneling, using SSL etc.

We will be using Amazon S3 to store our data. Amazon S3 can be used to store a huge amount of data at very low cost. Amazon Rekognition can be used to analyze images/videos. Through Amazon Rekognition we can add image and video analysis to our applications using deep learning. Amazon Rekognition provides highly accurate facial analysis. It has several advantages like low cost, and it provides highly scalable image and video analysis. Amazon Rekognition can be fully integrated with Amazon S3.

Amazon Rekognition retrieves metadata from images and videos. It has two API - one is for images and the other one is for videos. For using Amazon Rekognition, we first need to sign into an Amazon account, create a role, create S3 bucket and upload image/video. After uploading the image/video we can write our own function in any chosen language using functions such as json, boto3. The function must be defined in such a way that it detects the labels from that image/video. In the function we can also define MaxLabels and MinConfidence.

MaxLabels imposes a restriction on the max number of labels that can be retrieved from any image/video. MaxLabels is not a mandatory parameter though, if it is removed, we will get as many parameters that are present in that image. Once the function is written we can save it and then later test it. Testing the function will in turn generate new logs consisting of labels identified and confidence level.

Though Amazon Rekognition can also be used to analyze images/videos. Our primary focus is on building machine learning and deep learning models. Hence, we will upload the dataset consisting of videos first into S3 bucket and then SageMaker, the data will then be used to train machine learning models.

Training Deep Learning models requires even more data as compared to Machine Learning models. Data should not be class imbalanced otherwise the models will show bias towards the samples that are in the majority and hence it will affect the overall performance of the models and can also lead to overfitting. Data Augmentation is an efficient way to handle insufficient data. It helps in increasing the size of training data which in turn will help in avoiding overfitting of data by better generalizing the model.

We are planning to extract configural features from video frames by using tools like OpenCV, Gabor Filters etc. Through this we can then derive the feature vectors that define the shape and shading changes of action units in the face. These action units will then be evaluated for emotion recognition. Our aim is to perform comparative study between traditional machine learning models like SVM, Random Forest etc. with deep learning algorithms such as Convolutional Neural Networks (CNN), to recognize the emotion as well as the respective intensities of those emotions. The dataset that we are using is of students attending online lectures. Once their emotions are classified, we will be able to understand at what time the students are more focused and when their attention drifts from the lectures.

2.2 Project Development Methodology

In this project, the CRoss Industry Standard Process for Data Mining methodology, also called as CRISP-DM methodology is followed for project management. The CRISP-DM methodology helps phase out the project into components that can be considered as minimum

viable product considering various factors such as dependencies, timelines, and bandwidth. With this, we arrive at the most efficient effort and time estimates for each task in the project right from problem definition till deployment (Luna et al. 2021). As the first level, we can divide the project into six major phases and later add appropriate tasks and sub-tasks under each phase. The detail of each phase is explained in the following sections.

2.2.1 Business Understanding

The first phase of CRISP-DM is Business Understanding, which allows us to identify the context, scale, and intensity of the problem at hand, to be able to better define it. In today's world, every movement of us humans are recorded by sources that we may or may not know of. The majority of this recording is in the form of surveillance at public areas such as roads, transportation vehicles, stores, hospitals, prisons, etc. for the purpose of security. However, these videos have lately been utilized to analyze the behavior of people, identify patterns with it and make use of such patterns for decision making by various industries. Apart from surveillance, online platforms such as educational lectures, virtual apparel stores, gaming applications, etc. record human activities which are later used to understand the level of engagement they show. Irrespective of the nature of the recording, it is evident that multiple industries are trying to use such videos to track the level of engagement shown by people at different stages of these activities and use the insights derived for betterment of their application, store, collection, content and/or user platform. With emotion recognition having its base in a plethora of industries, and the rapid increase in the volume of data collected, it is important for industries to be prompt in identifying patterns effectively and using them at the right time for the right set of people.

Given the excessive industry applications, it is important that the data collection, cleaning and processing and the analysis are rapid and accurate. Advanced analytics tools and techniques for efficient data processing, implementation of appropriate machine learning and deep learning algorithms and automation of the same is therefore instrumental. Irrespective of the industry, it is important to determine the important frames from a long video, identify relevant features across different backgrounds and illumination, ensure an unbiased sample for modeling, pick the most appropriate machine/deep learning algorithms for emotion classification and detection of its intensity. Depending on the industry and the application, the requirements of sampling, illumination, and the categories of emotion to classify could vary. So, it is important to understand the objective and significance of each of these and set the requirements straight at this phase of the project.

In this project, we are focusing on engagement recognition of students attending an online course. Effective identification of emotion despite varied illumination throughout the video, across each student is a primary requirement. The emotions are to be classified into 6 categories on an intensity scale of four, as stated in the introduction. Ensuring a balanced sample, given different ratios of students from various diversities participate in the online lecture is an additional requirement.

2.2.2 Data Understanding

The next step in the CRISP-DM methodology is to understand the data. There needs to be clarity on the expectations of data required, its quality, and volume. For the problem in hand, videos from various sources are the input data, from which image frames are extracted to identify the emotion portrayed. In this project, as engagement recognition during educational lectures is chosen, the data will be each attendee's activity throughout the lecture. This data is procured

from DAiSEE, which consists of more than 9000 videos with varying quality, background and illumination procured by the authors of (Gupta et al., 2018). The videos are sourced from attendees of different ages, race, and location. These videos are converted into image frames, generating more than 2 million images.

As there is a possibility of the data scaling up, it is essential to extract selected images that significantly portray the different emotions of the attendees. The images are parsed through, and preliminarily categorized as the ones with high, medium, and low resolution. Each category will be separately trained by the model for achieving optimum accuracy. For the DAiSEE dataset considered, 99 percent of the image frames are of high resolution, so the remaining one percent could be considered outliers. Based on the initial exploration of the videos and corresponding image frames, the dataset is fairly consistent, proving it's of good quality as claimed by the authors of (Gupta et al., 2018).

2.2.3 Data Preparation

After getting a clear understanding of the Business and Data, the next step in CRISP-DM process is Data Preparation. This step entails cleaning and transforming the data to make it ready for modeling. For the purpose of modeling, it is instrumental to have the data in a particular format such as having no missing values, encoding the categorical data points into numerical, removing outliers, regularizing the data, extracting and feeding only the most significant features to minimize bias, etc. In our project, for the image frames extracted from videos, different feature extraction and selection approaches could be followed for different models. The data is first loaded onto Jupyter notebook using TensorFlow.

Each image is represented by a series of pixels, all numeric, the array of which is the input data for the models. For standard machine learning approaches like SVM, vectorized

feature extraction techniques are used. For deep learning based approaches such as ANN, CNN etc., feature selection approaches like Harris Corner detection, Speeded-Up Robust Features (SURF) and Features from Accelerated Segment Test (FAST) are used (Patel, 2020). Batch standardization, a method of preparing data for deep learning algorithms wherein the contributions to each layer are normalized for each group is used to standardize the data. Since noise can lead to performance concerns such as overfitting, it is removed from the dataset using Gaussian blur. The data is then sampled into train, test and validation datasets ensuring that the same distribution is maintained in all three (Pouya et al. ,2019).

2.2.4 Modeling

Once the data is prepped, we can use different machines and deep learning algorithms to develop the predictor. The first step is to shortlist the appropriate models that can be used for the kind of data and application we have. In this project traditional Machine Learning models such as SVM and kNN, and deep learning models such as ANN and CNN will be implemented. The features extracted using vectorization are used for machine learning models. The models are iterated with multiple values of alpha to arrive at the optimum value of weights.

The developed models are further optimized by tuning the hyperparameters. For the deep learning models, multiple layers need to be developed. A basic neural network model includes an input layer, one or more hidden layer(s) and an output layer. Convolutional neural networks have convolutional layers with 5x5 filters and 32 layers. Activation functions such as Sigmoid and Tanh are used to transform the results to a particular range like zero to one. Additional layers such as dropout and MaxPool2D could be used to reduce overfitting and transform the images to one-dimension. Once the models are trained, k-fold cross validation is done to further optimize

the models and ensure they aren’t overfit to the training data. Post validation, testing can be performed using the test dataset.

2.2.5 Evaluation

Based on their evaluation metric, the models generated during the modeling phase are compared to one another. For deployment, the model that performs the best in terms of the evaluation metrics for the decided business requirements is picked. By comparing the anticipated outputs to the original labeled data, evaluation metrics capture the performance of models. For classification algorithms, metrics such as Accuracy, Specificity, Sensitivity, and F1 score can be used to evaluate the collected performance. Mean Squared Error (MSE) and Region of Convergence (RoC) curves are also generated as regression metrics. An evaluation report is provided after the performance of machine learning models is compared to that of deep learning models. For testing and deployment, the model with the best training and validation accuracy is chosen.

2.2.6 Deployment

According to CRISP-DM, the project's final stage is deployment. AWS is used for deployment in this project. used to deploy the model that was chosen during the evaluation. An automated pipeline is created using Jenkins pipeline to read input data at the required frequency (daily/weekly/monthly etc.), run the preprocessing and modeling framework on the new data, and store the prediction results along with the performance report to the designated cloud storage repository, which is AWS in this case. The relevant stakeholders are then sent an email with the location of the results and the performance reports. To trace any decline in performance, the deployed models must be regularly checked. It's critical to re-train and analyze deployed models on a regular basis to guarantee they're up to date with new data. Receiving feedback and

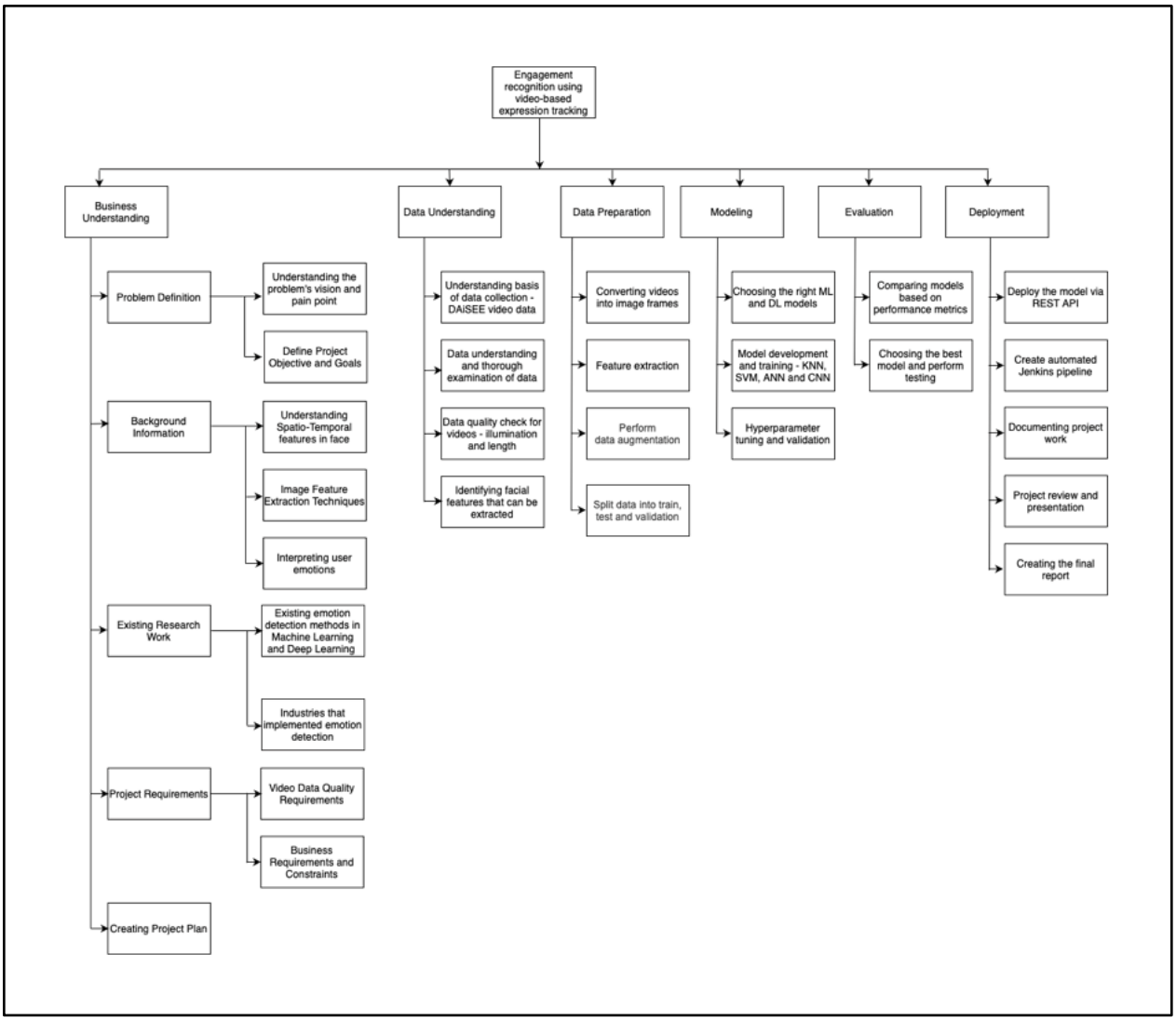
recommendations from users aids in improving the model's accuracy. An extensive documentation in the form of report and presentation is done.

2.3 Project Organization Plan

CRISP-DM is a commonly used industry standard technique for analytics models. In order to establish the work breakdown structure in our project, we followed the same process. There are six distinct stages, as shown in Figure 1.

Figure 1

Work breakdown structure



Note. Work breakdown structure illustrating the 6 CRISP-DM phases

Figure 1 represents the WBS that we created for our project. It represents the 6 CRISP-DM phases in detail.

We'll look at the linear model and take a quick look at the business and its proposal in the first step, business knowledge. True company needs, objectives, and deadlines are identified, as well as a deeper dive into a given sector. We'll be determining a user's emotion in this section. The second step, data comprehension, focuses on the breadth and depth of accessible data in order to assess the model's correctness.

We will utilize the DAiSEE dataset, which contains 9068 video clips gathered from 112 people between the ages of 18 and 30 who are of Asian origin. (Gupta and colleagues, 2018). We will need to sort out the unnecessary trash data in the third phase, data preparation, by cleaning, formatting, and deleting duplicate and unwanted data. The backdrop picture and data can be muted because we will be working with facial data. We may also use machine learning algorithms to eliminate unnecessary characteristics and focus just on specific regions of the face, which are crucial in evaluating a user's sentiment. We can save the model's additional processing time this way. The data is known to perform better with less essential characteristics and cleansed data.

We will train the data with the finest state-of-the-art deep learning models like convoluted neural networks (CNN), artificial neural networks (ANN), and a few Machines Learning models like support vector machines in the fourth step, modeling. Hyperparameter adjustment may be done based on the performance of the models.

In the fifth step, we will check the veracity of our model based on accuracy, recall, F1 score, confusion matrix, and area under the curve. Later, the best model will be chosen based on the performance and perform testing on that model.

Sixth and the last phase is deployment which includes recording project activity, conducting a project review, and eventually generating a report. The actual model will be deployed using the REST API and automated using the Jenkins pipeline.

2.4 Project Resource Requirements & Plan

The project is implemented using cloud technology AWS services, Python Jupyter notebook and visualization tools. For the AWS cloud services to work efficiently, we need a local machine with eight GB RAM or above, Windows 64-bit processor and a minimum of one

GB graphic card. AWS S3 is a cloud storage service used to store dataset and other project related files. Python Jupyter Notebook version 3.7 is used to perform data preprocessing and cleansing. Tableau Desktop software and python Jupyter notebook are used for data visualization and statistical analysis. Machine learning algorithms to build, train, test and deploy the models are implemented using AWS Sagemaker. It is a cloud - based service that handles the entire machine learning process flow, from labeling and preparing your data to selecting an algorithm, training, tuning, and optimizing model for deployment, predicting outcomes, and implementing solutions in a production-like environment. Machine learning frameworks - Tensorflow, pytorch, XGboost supported by AWS Sagemaker are used to perform complex modeling implementations like CNN and other requirements. In order to achieve end to end project solution utilizing various AWS services, AWS CLI is installed on the local machine to provide a single integrated platform for interaction and access to various AWS services used in the project.

Table 1

Resources and Cost Estimation

Utility	**Resource Type**	**Tool/Application**	**Duration**	**Cost Estimation**
Cloud Service Management Tool	Software	AWS CLI	2 Months	Free
Data Storage	Hardware	AWS S3	2 Months	$30
Local Machine	Hardware	64-bit version machine or later	2 Months	$800
Visualization tool	Software	Tableau Desktop	2 Months (Student License)	Free

Machine Learning Modeling	Software	AWS Sagemaker	2 months	$120
Machine Learning Frameworks	Software	Scikit-learn, TensorFlow	2 months	Free
Data Pre-Processing	Software	python Jupyter notebook	2 months	Free

Note. All resources and cost estimations are as per project requirements

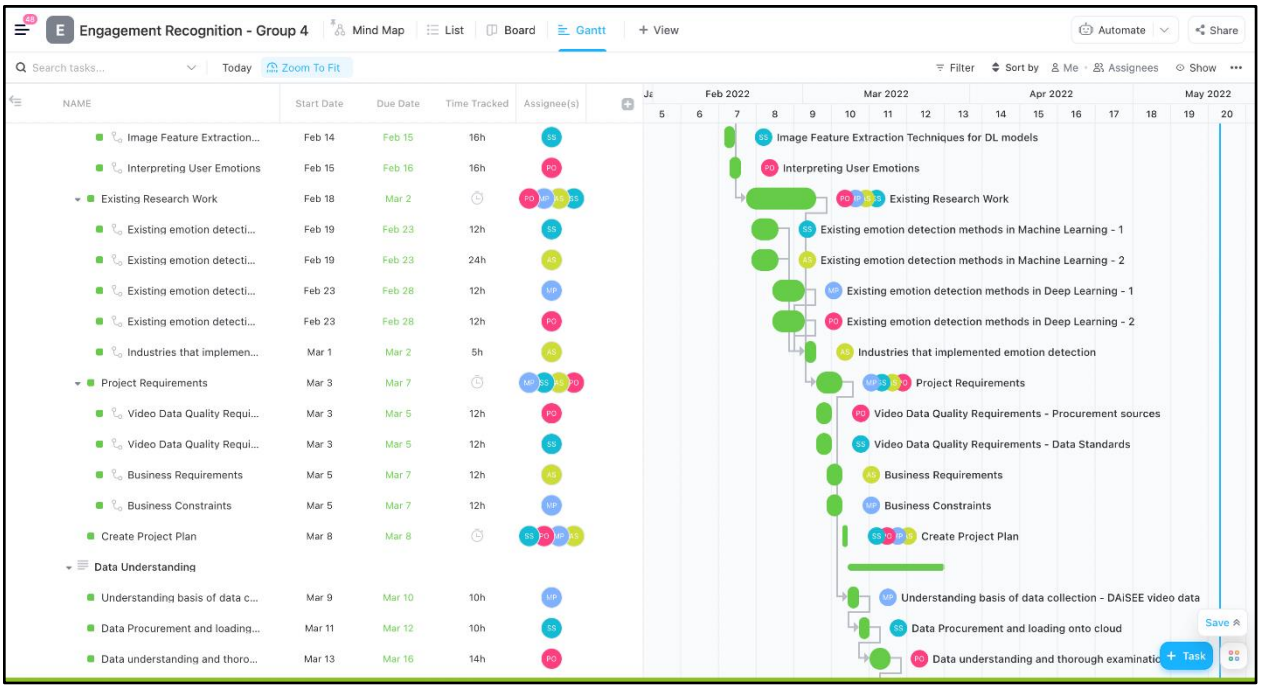
Table 1 shows all the resources and cost estimates for our project, so in order to implement this project in real time these are the resources that will be required, and the cost associated with it.

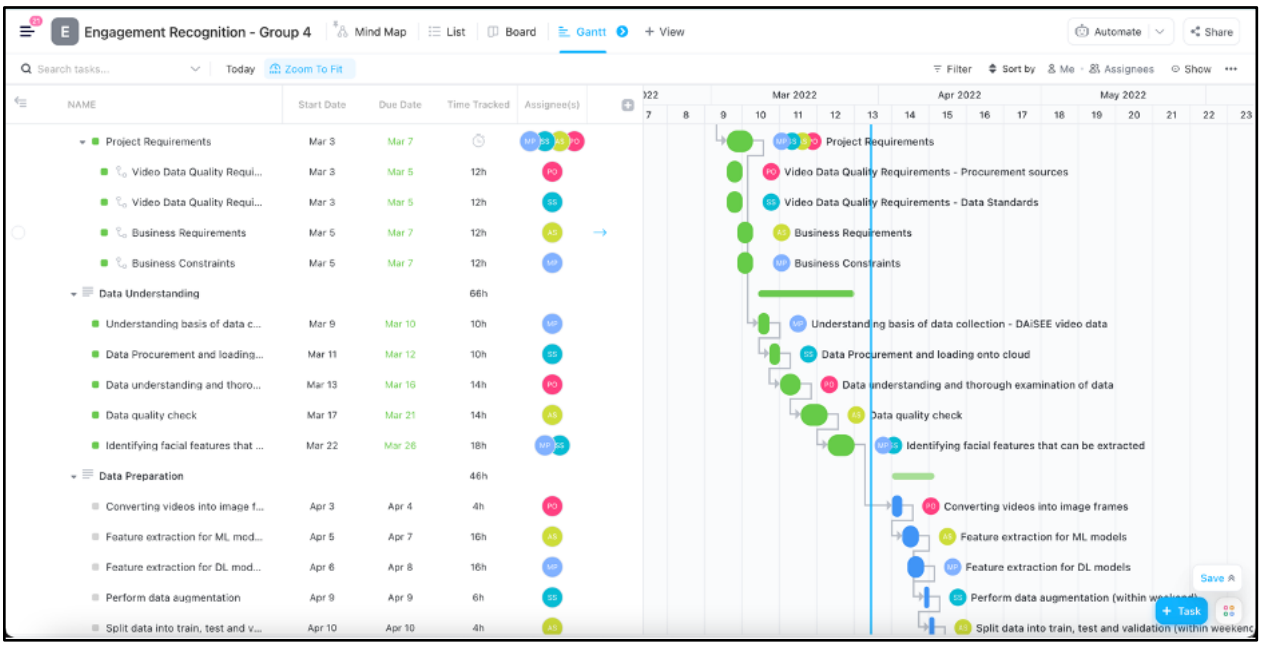
2.5 Project Schedule

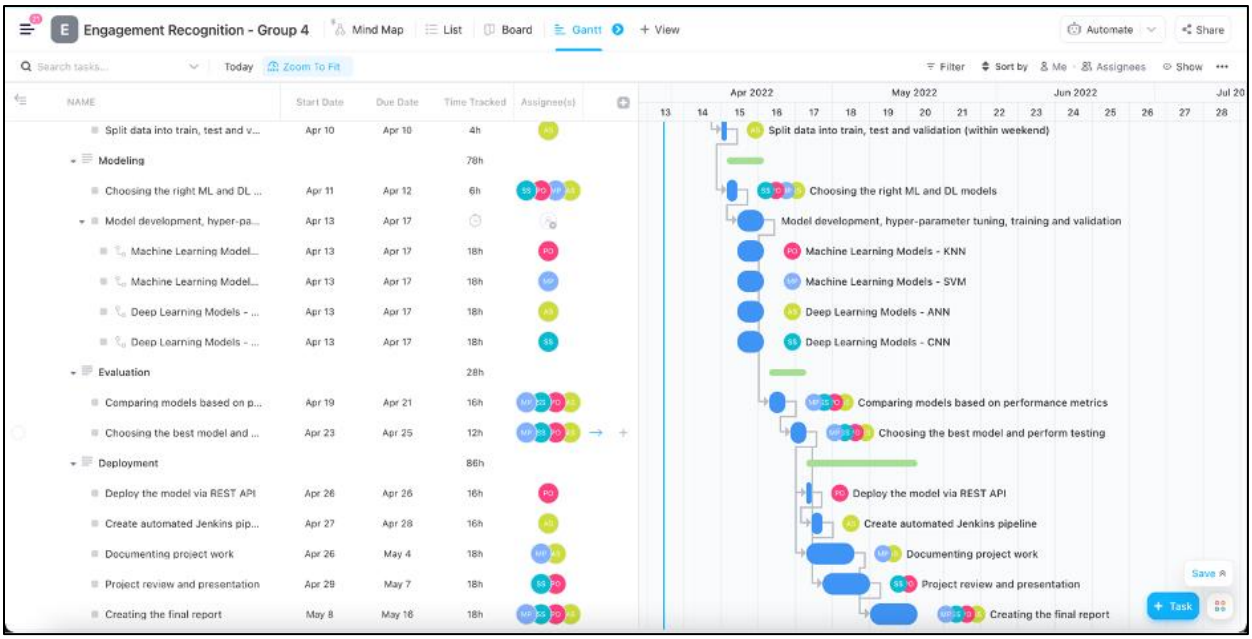
Gantt chart helps us understand what tasks need to be done at any particular time by whom and if there's any dependency for completing the same. The chart portrays the project schedule with each task along with its start and due date, time estimate to complete it and team member(s) that are accountable for its completion (Bednjanec & Tretinjak, 2013).

Figure 2

Gantt Chart


Engagement Recognition - Group 4
Mind Map
List
Board
Gantt
+ View
Automate
Share
Search tasks...
Today
Zoom To Fit
Filter
Sort by
Me
Assignees
Show
NAME
Start Date
Due Date
Time Tracked
Assignee(s)
Feb 2022
Mar 2022
Apr 2022
May 2022
Image Feature Extraction... Feb 14 Feb 15 16h
Image Feature Extraction Techniques for DL models
Interpreting User Emotions Feb 15 Feb 16 16h
Interpreting User Emotions
Existing Research Work Feb 18 Mar 2
Existing Research Work
Existing emotion detecti... Feb 19 Feb 23 12h
Existing emotion detection methods in Machine Learning - 1
Existing emotion detecti... Feb 19 Feb 23 24h
Existing emotion detection methods in Machine Learning - 2
Existing emotion detecti... Feb 23 Feb 28 12h
Existing emotion detection methods in Deep Learning - 1
Existing emotion detecti... Feb 23 Feb 28 12h
Existing emotion detection methods in Deep Learning - 2
Industries that implemen... Mar 1 Mar 2 5h
Industries that implemented emotion detection
Project Requirements Mar 3 Mar 7
Project Requirements
Video Data Quality Requi... Mar 3 Mar 5 12h
Video Data Quality Requirements - Procurement sources
Video Data Quality Requi... Mar 3 Mar 5 12h
Video Data Quality Requirements - Data Standards
Business Requirements Mar 5 Mar 7 12h
Business Requirements
Business Constraints Mar 5 Mar 7 12h
Business Constraints
Create Project Plan Mar 8 Mar 8
Create Project Plan
Data Understanding
Understanding basis of data c... Mar 9 Mar 10 10h
Understanding basis of data collection - DAiSEE video data
Data Procurement and loading... Mar 11 Mar 12 10h
Data Procurement and loading onto cloud
Data understanding and thoro... Mar 13 Mar 16 14h
Data understanding and thorough examinatio
Save
+ Task


Engagement Recognition - Group 4
Mind Map
List
Board
Gantt
+ View
Automate
Share
Search tasks...
Today
Zoom To Fit
Filter
Sort by
Me
Assignees
Show
NAME
Start Date
Due Date
Time Tracked
Assignee(s)
Mar 2022
Apr 2022
May 2022
Project Requirements Mar 3 Mar 7
Project Requirements
Video Data Quality Requi... Mar 3 Mar 5 12h
Video Data Quality Requirements - Procurement sources
Video Data Quality Requi... Mar 3 Mar 5 12h
Video Data Quality Requirements - Data Standards
Business Requirements Mar 5 Mar 7 12h
Business Requirements
Business Constraints Mar 5 Mar 7 12h
Business Constraints
Data Understanding 66h
Understanding basis of data c... Mar 9 Mar 10 10h
Understanding basis of data collection - DAiSEE video data
Data Procurement and loading... Mar 11 Mar 12 10h
Data Procurement and loading onto cloud
Data understanding and thoro... Mar 13 Mar 16 14h
Data understanding and thorough examination of data
Data quality check Mar 17 Mar 21 14h
Data quality check
Identifying facial features that ... Mar 22 Mar 26 18h
Identifying facial features that can be extracted
Data Preparation 46h
Converting videos into image f... Apr 3 Apr 4 4h
Converting videos into image frames
Feature extraction for ML mod... Apr 5 Apr 7 16h
Feature extraction for ML models
Feature extraction for DL mod... Apr 8 Apr 8 16h
Feature extraction for DL models
Perform data augmentation Apr 9 Apr 9 6h
Perform data augmentation (within
Split data into train, test and v... Apr 10 Apr 10 4h
Split data into train, test and validation (within weekend
Save
+ Task



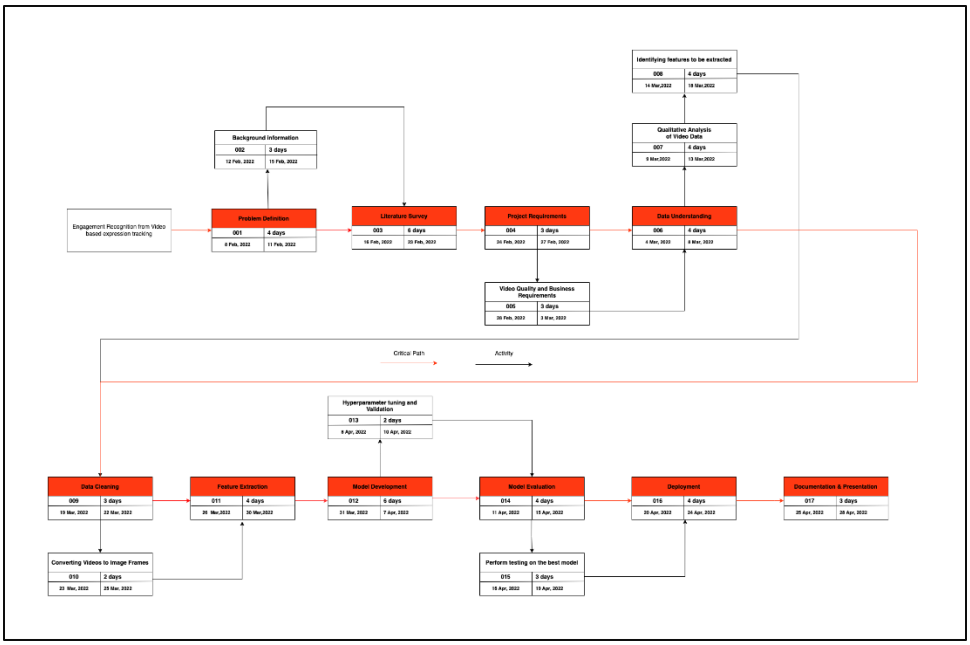
Note. Gantt chart illustrating tasks, timelines and status

Figure 2 represents the Gantt chart that we created for our project. Gantt chart is used to estimate duration of the project in total and identify the project's pain points and priority tasks.

The PERT chart helps in determining the shortest time needed to accomplish all tasks by finding the project's critical path. The critical path is shown by red arrows, and the critical tasks are indicated by red boxes.

Figure 3

PERT Chart



Note. Pert chart with tasks, dependencies and critical path

Figure 3 represents the PERT chart that we created for our project. We are using it to monitor our project tasks and deadlines.

3. Data Engineering

3.1 Data Process

We need to record videos of students attending online lectures to collect data for this project. We will use cameras with various resolutions to record the footage. The lighting in all these videos might be different. To counteract this, we will use image normalization to maintain a consistent contrast across all the images. Many other types of transformations will be used which as well is detailed in the later sub-chapters. So, while collecting data, our focus will be to get the subject in the frame and to capture the emotions. Everything else that comes along as the noise or inconsistent data can be removed or transformed. The exact specifications and the way the data will be collected are mentioned in the next section. The collected data needs to be carefully annotated by their respective emotions. In all the expected data volume would be around 8000 videos for training, 1000 videos for validation and 1000 videos for testing, each of which would be 10 seconds long. The data would take up around 15gb of storage space, which will be saved in google drive for easy access. Each video is recorded for 10 seconds with 30 frames per second.

The initial step is to collect all the recorded videos and later do the transformations one by one. The second step would be to identify the missing data and inconsistencies. This includes frames having blurred images and no images at all. This is being done to maintain consistent data for the algorithms to compute faster with good accuracy. The third step would be to eliminate all the duplicates. This step is being done because the subject might not move for a long duration and might sit in the same position and posture and with the same emotion. We eliminate duplicate frames by identifying frames having a similarity score of 85% and above. This step is briefed in the data pre-processing section. The pre-final step is to maintain a constant brightness

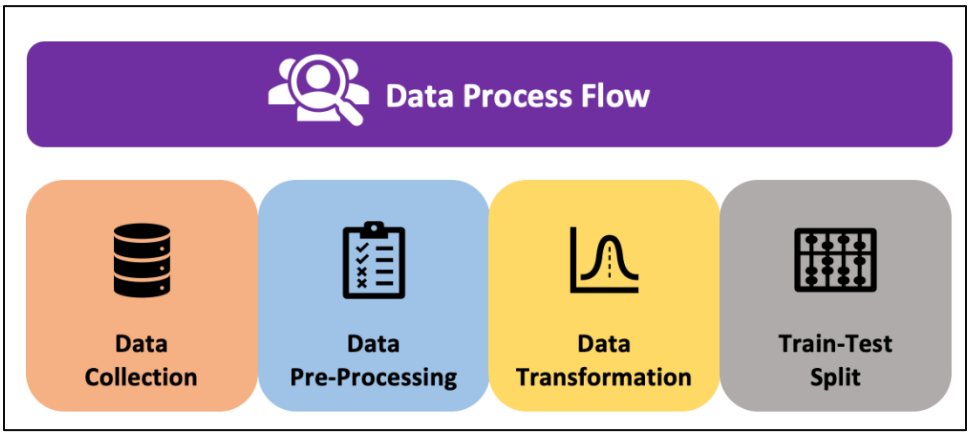
and contrast in all the images. This is achieved by doing image transformations which include transformations like image smoothing, image normalization, grey scale imaging is used for faster training of the data, resizing which is used to bring all the images to correct dimensions and give more examples and diversity to the training dataset, and a myriad of other transformations is performed. These transformations are further briefed in the data transformation section of this chapter.

Finally, the data is shuffled to maintain consistency and they are split into training, testing and validation datasets. The model will be trained on the training dataset, tested on the training dataset and validated on the validation dataset.

The entire data process flow from data collection to preparation is depicted in Figure 11.

Figure 4

Data Process Diagram



Note. Data Process Diagram

3.2 Data Collection

Our problem statement is engagement recognition from videos using facial expression tracking. In this project, we are planning to use videos of students attending lectures and gauge their expressions throughout the class.

The parameters and data quality requirements we expect for effective prediction include the following:

1. Students of different diversity need to be considered.
2. There should be a reasonable variation in the background from which students attend the lecture.
3. Different lighting settings with varying brightness and contrast for the model to train on.
4. Only one student should be the primary subject of each video.
5. Students should portray only one expression in every 10 second periods, to ensure clear distinction between expressions for training.
6. Only student video will be captured, and not their audio.
7. Videos should be captured from cameras of varying resolutions.
8. The camera capturing the video should be placed in a stable setup, without a lot of movements.
9. All 6 expressions of each student need to be recorded individually.
10. At least 100 participants need to be recorded, to ensure enough variety for the training set.
11. There should be 20 videos for each expression from each participant, totaling 80 videos from each participant.

For collecting the data, we need to ensure the equipment setup and processing requirements are planned and executed appropriately. For this project, the data collection process involves the following:

1. Collection of lecture videos from each student occurs through a dedicated workstation, operating under Mac OS.

2. The workstation should have a video capture card to capture the contents of the screen and later encoded as a high-quality video file.
3. A script is used to record the video whenever a meeting starts so that the emotions can be recorded.
4. All the files captured are stored in .mov format.
5. The files are uploaded to google drive and it is only accessible to our group members, approved researchers who have taken prior permission and other prospects who abide by the privacy laws.
6. Those who are interested can do so by requesting the dataset through a google form link explaining the need and use case of the research.
7. The data provided does not have any labels and they need to be annotated later according to the respective emotions.
8. The requested data information of users is collected for statistical analysis. Username, age, research interests and other vital information will be collected to provide the dataset.

For this project, based on the above requirements set, we've chosen to work with the DAiSEE dataset (Gupta et al.), for collecting this dataset, HD web camera was used. The camera was focused on students who were watching online lecture videos followed by recreational videos. To stimulate real-world, environment videos were shown on a custom application. Changes in the engagement level of students were monitored. Total number of students were 112, who were pursuing various degrees from a university and through the recorded videos their emotions were labelled as frustrated, engaged, bored, or bewildered on a four-level intensity scale for the purpose of determining user involvement. This dataset contains 2,723,882 images extracted against 9068 clips and 25 hours of video, and this is large enough to train some of the best deep

learning models. Videos were recorded in different lighting conditions. A unique id was allocated to these recorded videos. CrowdFlower and Vote aggregation was used to label the videos. CrowdFlower avoids the error caused by bot labelling. The dataset includes 3 folders that correspond to Train, Validation and Test data. The Train and Validation data are annotated, while the Test data doesn't have labels.

Table 2

Data Collection Plan

What?	Variable title	1
	Input (X) or output (Y) variable?	
	Unit of measurement	Pixels(1280x720)
	Data type	JPG
	Collection method	HD web camera (1920x1080, 30 fps, focal length 3.6mm, 78◦ field of view)
	If manual	NA
MSA	Gauge/instrument	Camera (1920x1080, 30 fps, focal length 3.6mm, 78◦ field of view)
	Location	Dorm rooms, crowded lab spaces, library
	Gauge calibrated?	Yes
	Measurement system checked?	Yes
	Precision (R&R) adequate?	Yes
	Accuracy adequate?	Yes
Historical data	Historical data exist?	No
	Source of historical data	NA
	Historical data representative/reliable?	NA
	Mean	NA
	Upper specification limit	NA
	Lower specification limit	NA
	Standard deviation	NA
	Target	NA
Sampling	Minimum sample size (MSS)	8000 videos
	Sampling frequency	Yes
	Sub-grouping needed?	0.2
	Sub-group size	1000
	Stratification needed? (time, shift)	NA
Who?	Data collector	Analyst
	Operational definition exist?	Yes
	Data collector trained?	Yes
	Resources available for data collector?	Yes
When?	Start date	44593
	Due date	44647
	Duration (in days)	54 days

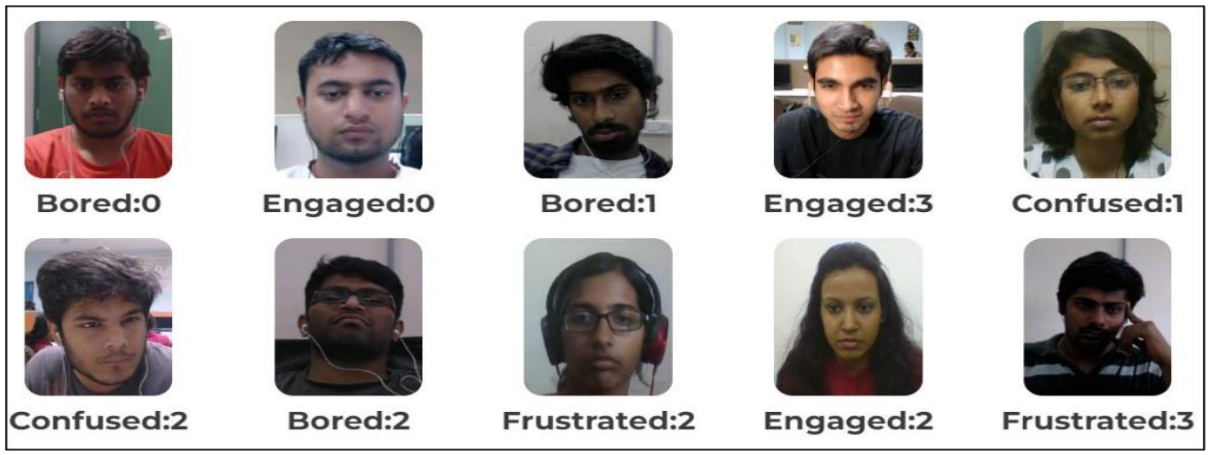
Note. Data Collection Plan

The Data Collection Plan of our Project is mentioned in detail in Table 2.

Below clips are taken from videos captured in DAiSEE dataset. The dataset is one of a kind to perform multilabel engagement classification. The labels represent the expression portrayed along with the intensity level of the expression.

Figure 5

Samples from raw dataset



Note. Samples of raw data from DAiSEE dataset

Figure 5 and Figure 6 represent the samples of raw data from DAiSEE dataset that we used in our project.

DAiSEE dataset also consists of .csv files that have information related to the engagement levels of videos, one such label information from the training dataset csv file is shown below.

Figure 6

Sample showing engagement level of a student

ClipID	Boredom	Engagement	Confusion	Frustration
1100011002.	0	2	0	0
1100011003.	0	2	0	0
1100011004.	0	3	0	0
1100011005.	0	3	0	0
1100011006.	0	3	0	0
1100011007.	1	2	0	0
1100011008.	0	3	0	0
1100011009.	0	2	1	0
1100011010.	0	3	0	0
1100011011.	0	3	0	0
1100011012.	0	2	2	0
1100011013.	0	3	0	0
1100011014.	0	3	0	0

Note. Sample showing engagement level of a student

3.3 Data Pre-Processing

The next step in Data Engineering is Data Pre-processing. This includes steps like converting videos to images, removing duplicates, identifying missing values, handling noisy and inconsistent data points, and performing exploratory data analysis. It is one of the most instrumental steps that has a direct impact on the machine learning models' performance.

Since the inputs are videos, our first step is converting the videos to image frames. Each video in our dataset is about 10 seconds long and at 30 fps. So, converting them will generate 300 images per video. To do the conversion, we use the check output function from the subprocess library. The function will generate image frames in .jpg format for each video input and save them in the directory mentioned. The dataset already contains data split into Train, test and Validation folders. We have 1,887 videos from 21 users, generating 566,100 image frames in the Test dataset. In the Training dataset, we have 5,552 videos from 70 users, generating 1,665,600 image frames. For the validation dataset, we have 1,742 videos from 22 users, generating 522,600 image frames. Overall, we have 2,720,400 image frames generated from

9,068 video clips. Figures 7, 8, 9. and 10 are snaps of the conversion codes, images generated and counts of the same.

Figure 7

Function to split videos into image frames

```
[16] # Split a video and save each frame
     def split_video(video_file, image_name_prefix, destination_path):
         return subprocess.check_output('ffmpeg -i "' + destination_path+video_file + '" ' + image_name_prefix + '%d.jpg -hide_banner', shell=True, cwd=destination_path)
```

Note. Python code defining function that splits videos into image frames

Figure 8

Calling split_video function

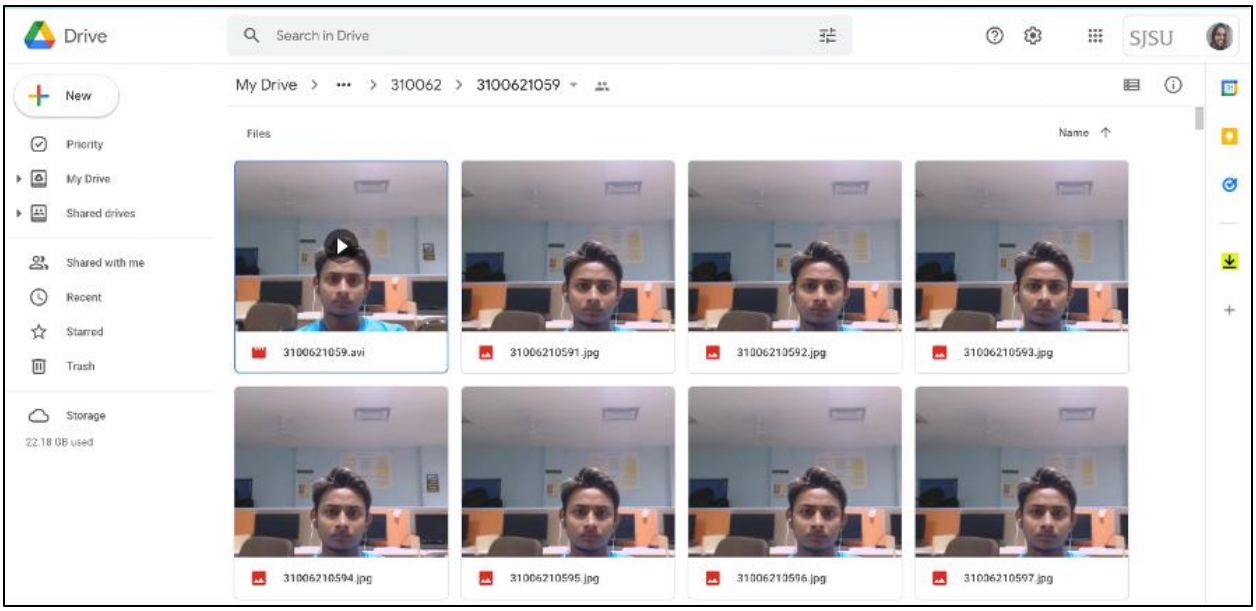
```
# Code for getting frame from each video
for ttv in ['Train', 'Test', 'Validation']:
    print('ttv is', ttv)
    if ttv == '.DS_Store':
        continue
    users = os.listdir('/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/')
    for user in users:
        print('user is', user)
        if user == '.DS_Store':
            continue
        currUser = os.listdir('/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/')
        for extract in currUser:
            print('extract is', extract)
            if extract == '.DS_Store':
                continue
            clip = os.listdir('/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/')[0]
            if clip == '.DS_Store':
                clip = os.listdir('DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/')[1]
            print (clip[:-4])
            path = os.path.abspath('.')+'/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/'
            split_video(clip, clip[:-4], path)
            # break
```

```
ttv is Train
user is 110013
extract is 1100131011
1100131011
extract is 1100131009
1100131009
extract is 1100131006
1100131006
extract is 1100131010
1100131010
extract is 1100131007
1100131007
extract is 1100131017
1100131017
extract is 1100131019
1100131019
extract is 1100131012
1100131012
user is 310070
extract is 3100701021
3100701021
extract is 3100701024
3100701024
extract is 3100702051
3100702051
extract is 3100701031
3100701031
```

Note. Iterating through train, test and validation folders to call split_video function

Figure 9

Generated Image Frames



Note. Generated Image Frames for train, test and validation datasets available in the drive

Figure 10

Count of videos and image frames

```
Test dataset:
 Total number of Users: 21
 Total number of videos: 1887
 Total number of image frames: 566100

Train dataset:
 Total number of Users: 70
 Total number of videos: 5552
 Total number of image frames: 1665600

Validation dataset:
 Total number of Users: 22
 Total number of videos: 1742
 Total number of image frames: 522600

Overall - Total number of videos: 9068
Overall - Total number of image frames: 2720400
```

Note. Count of videos and generated image frames for train, test, and validation datasets

With over 2.7 million images being generated from videos, there is a high chance that multiple image frames are identical as students tend to sit in the same posture for seconds together. So, for this dataset, removing identical (or duplicate) frames is an important data

cleaning step that could save a lot of computational time, as shown in Figure 11. To remove duplicates, we are using the find_duplicates_to_remove function in the method_object object of DHash library with a threshold of 85% similarity between images. We can see from Figure 12 that after removing duplicates, we have a total of 190,428 image frames remaining.

Figure 11

Removing duplicate image frames

```
image = Image.open('/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/'+frame+'')
method_object.find_duplicates_to_remove(image_dir='/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/', min_similarity_threshold=0.85)

Found 270 images with one or more duplicate/similar images in 13.2149 seconds.
{'8264120101.jpg': {'duplicates': ['/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/8264120102.jpg'],
  'location': '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/8264120101.jpg'},
 '82641201010.jpg': {'duplicates': ['/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/82641201011.jpg'],
  'location': '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/82641201010.jpg'},
 '826412010100.jpg': {'duplicates': ['/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010101.jpg',
   '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010102.jpg',
   '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010103.jpg'],
  'location': '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010100.jpg'},
 '826412010101.jpg': {'duplicates': ['/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010102.jpg',
   '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010103.jpg',
   '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010104.jpg'],
  'location': '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010101.jpg'},
 '826412010102.jpg': {'duplicates': ['/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010103.jpg',
   '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010104.jpg',
   '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010105.jpg'],
  'location': '/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Test/826412/826412010/826412010102.jpg'},
```

Note. Removing duplicate image frames with 85% as similarity threshold

Figure 12

Count of images after removing duplicates

```
Test dataset:
 Total number of Users: 21
 Total number of videos: 1887
 Total number of image frames: 39627

Train dataset:
 Total number of Users: 70
 Total number of videos: 5552
 Total number of image frames: 116592

Validation dataset:
 Total number of Users: 22
 Total number of videos: 1742
 Total number of image frames: 36582

Overall - Total number of videos: 9068
Overall - Total number of image frames: 190428
```

Note. Count of images in Train, Test and Validation datasets after removing duplicate frames

Once the duplicates are removed, the next step is to identify if there is any incomplete or missing data. Incomplete or missing data for our project would be image frames where there is

no active student present. Recognizing such cases from image frames automatically itself will turn out to be an object detection problem. In this project, we are hence manually removing image frames that do not have an active student that is attending the lecture. We manually parse through the 190,428 image frames that remain after removing duplicates to see if there is a student as primary subject in the image. All image frames without a student as primary subject are removed. In the DAiSEE dataset that is used in this project, only a handful of such cases were present. The number of image frames after removing missing data is 190,416. Such manual identification, however, might not be the best method to identify missing records. To automate this process, object detection algorithms which identify the presence of a primary subject using bound boxes can be used. Multiple frameworks such as Facebook's Detectron and Detectron 2, OpenCV, etc. Which use Deep learning methods such as CNN and RCNN can be directly used to determine the bounding box of the subject and use the pixel values withing the bounding box to confirm the presence and absence of a student.

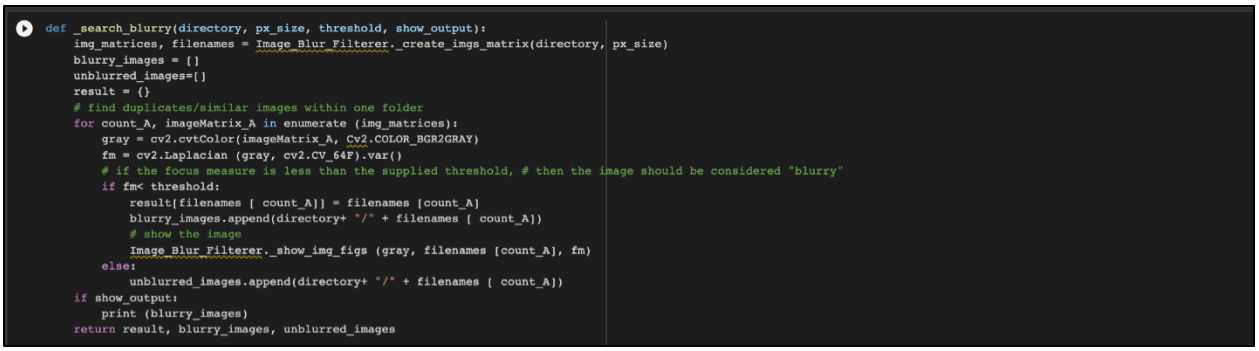
Converting videos to image frames potentially means that every movement by a student, say turning the head slightly, will be captured in a detailed manner over a series of frames. In such cases, the frames that correspond to movement have higher chances to be blurred. Additionally, certain images can be blurred owing to the quality of the capturing device. It is also possible that images are corrupted due to technical issues. Such blurred images correspond to a higher level of noise and having them in the training set would potentially complicate the model and reduce its performance. Hence, images that are blurred beyond a certain extent are removed from the training data.

In the script shown in Figure 13, we compare the image's Laplacian variance to the limit set in the method's argument. If the value is less than the limit, the image is considered blurred.

The limit for our data has been chosen as 0.8, as shown in Figure 14. Figure 15 shows a sample blurred and unblurred image. After removing such noisy images, we can see from Figure 16 that 181,360 image frames remain.

Figure 13

Identifying and removing blurred images



```
def _search_blurry(directory, px_size, threshold, show_output):
    img_matrices, filenames = Image_Blur_Filterer._create_imgs_matrix(directory, px_size)
    blurry_images = []
    unblurred_images=[]
    result = {}
    # find duplicates/similar images within one folder
    for count_A, imageMatrix_A in enumerate (img_matrices):
        gray = cv2.cvtColor(imageMatrix_A, Cv2.COLOR_BGR2GRAY)
        fm = cv2.Laplacian (gray, cv2.CV_64F).var()
        # if the focus measure is less than the supplied threshold, # then the image should be considered "blurry"
        if fm< threshold:
            result[filenames [ count_A]] = filenames [count_A]
            blurry_images.append(directory+ "/" + filenames [ count_A])
            # show the image
            Image_Blur_Filterer._show_img_figs (gray, filenames [count_A], fm)
        else:
            unblurred_images.append(directory+ "/" + filenames [ count_A])
    if show_output:
        print (blurry_images)
    return result, blurry_images, unblurred_images
```

Note. Function to identify and remove blurred images

Figure 14

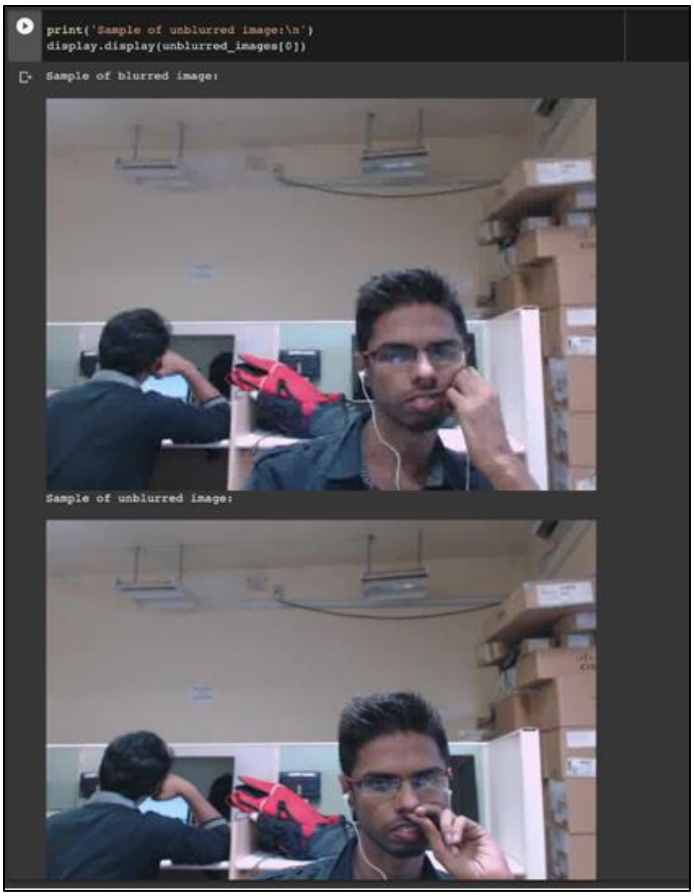
Function call to remove blurred images

```
            image = Image.open('/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/'+frame+'')
            method_object.find_duplicates_to_remove(image_dir='/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/', min_similarity_threshold=0.85)
            result, blurry_images, unblurred_images =_search_blurry('/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/'+ttv+'/'+user+'/'+extract+'/', 520, 0.8, False)

print('Sample of blurred image:\n')
display.display(blurry_images[0])

print('Sample of unblurred image:\n')
display.display(unblurred_images[0])
```

Note. Identifying and removing blurred images with a threshold of 0.8

Figure 15

Sample of blurred and unblurred image



Note. Sample of blurred and unblurred image

Figure 16

Count of images after removing blurred frames

```
Test dataset:
 Total number of Users: 21
 Total number of videos: 1887
 Total number of image frames: 37740

Train dataset:
 Total number of Users: 70
 Total number of videos: 5552
 Total number of image frames: 111040

Validation dataset:
 Total number of Users: 22
 Total number of videos: 1742
 Total number of image frames: 34840

Overall - Total number of videos: 9068
Overall - Total number of image frames: 181360
```

Note. Count of images after removing blurred frames

In video datasets, it is possible that certain frames do not have clear images of the subject.

For example, in one of the videos, for a few seconds, the entire face of the student is not seen, and in another case, there is an object hiding the face of the student in a few frames. However, the remaining parts of the video are still valid and can be taken into consideration. It is important to identify these inconsistencies and remove them to avoid feeding noise to the model. However, automatically identifying such inconsistencies will again be an object detection problem by itself and is not in the scope of this project. In this project, to avoid feeding these inconsistent data points to the model, we are manually identifying and removing them from the training set. Samples of inconsistent data frames are shown in Figures 17 and 18. In the DAiSEE dataset, only a handful of such frames were present, and after removing them 181,346 image frames remain in total, as shown in Figure 19. Like missing values, identifying and removing inconsistent frames can be automated using readily available object detection frameworks such as Detectron and OpenCV for larger datasets.

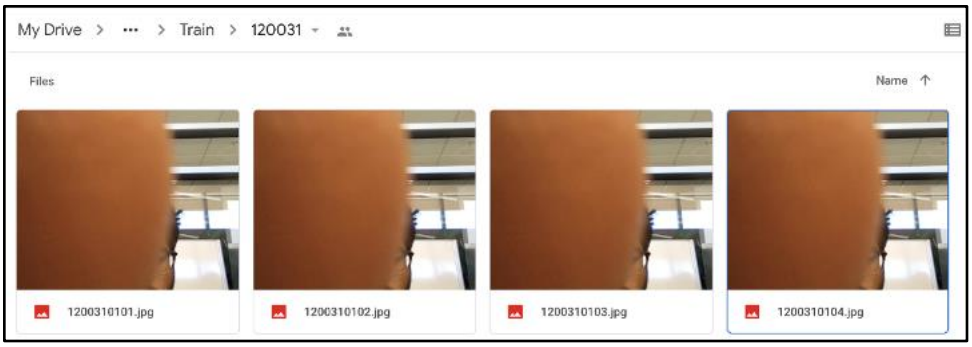
Figure 17

Samples of inconsistent data - 1

Note. Samples of inconsistent data – 1

Figure 18

Samples of inconsistent data - 2



Note. Samples of inconsistent data – 2

Figure 19

Count of images after removing inconsistent data

```
Test dataset:
 Total number of Users: 21
 Total number of videos: 1887
 Total number of image frames: 37740

Train dataset:
 Total number of Users: 70
 Total number of videos: 7552
 Total number of image frames: 111026

Validation dataset:
 Total number of Users: 22
 Total number of videos: 1742
 Total number of image frames: 34840

Overall - Total number of videos: 9068
Overall - Total number of image frames: 181346
```

Note. Count of images after removing inconsistent data

Once the data is cleaned, it is important to perform exploratory data analysis to understand the distribution and variety of data available. Data exploration is one of the most important steps in data analysis. Data Scientist and Data Analysts perform extensive EDA to discover hidden insights or patterns/trends from the data. These insights can then be used to build predictive models. A huge chunk of data science projects is usually allocated for data exploration since it also involves data cleaning and data preprocessing. EDA is a broader umbrella under which we

will first perform data cleaning like finding out the missing values, understanding data, finding out the shape of dataset, categorizing values, checking data type, summary statistics and then perform data preprocessing in which we will encode the categorical data, standardize the data, perform dimensionality reduction using PCA, LDA, SVD etc.

It is important for the model to predict expressions irrespective of the brightness of images. To understand the range of brightness in the input image frames, we compare the average brightness of each image and see the difference in the images that have the maximum and minimum brightness, as shown in Figure 20. If the difference is significant, we will have to normalize the brightness for unbiased prediction. From the results, we see that there is a difference of around 0.3 units in the average brightness of images, as shown in Figure 21. To ensure optimal prediction, we will normalize the brightness of image frames.

Figure 20

Exploration of brightness levels across image frames

```
def calculate_brightness(image):
    greyscale_image = image.convert('L')
    histogram = greyscale_image.histogram()
    pixels = sum(histogram)
    brightness = scale = len(histogram)
    for index in range(0, scale):
        ratio = histogram[index] / pixels
        brightness += ratio * (-scale + index)
    return 1 if brightness == 255 else brightness / scale
i=0
brightness= [0 for i in range(300)]
currUser = os.listdir('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/110014/1100141050/')
for extract in currUser:
  if extract == '1100141050.avi':
    continue
  image = Image.open('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/110014/1100141050/'+extract+'')
  brightness[i]=calculate_brightness(image)
  print('Brightness of %s is: %s' % (extract, brightness[i]))
  i+=1
max_value_1 = max(brightness)
min_value_1 = min(brightness)
avg_value_1 = 0 if len(brightness) == 0 else sum(brightness)/len(brightness)
currUser = os.listdir('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/310062/3100621059/')
for extract in currUser:
  if extract == '3100621059.avi':
    continue
  image = Image.open('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/310062/3100621059/'+extract+'')
  brightness[i]=calculate_brightness(image)
  print('Brightness of %s is: %s' % (extract, brightness[i]))
  i+=1
max_value_2 = max(brightness)
min_value_2 = min(brightness)
avg_value_2 = 0 if len(brightness) == 0 else sum(brightness)/len(brightness)
print('Maximum brightness: %s', max(max_value_1, max_value_2))
print('Minimum brightness: %s', min(min_value_1, min_value_2))
```

Note. Exploration of brightness levels across image frames

Figure 21

Samples of brightness levels across image frames

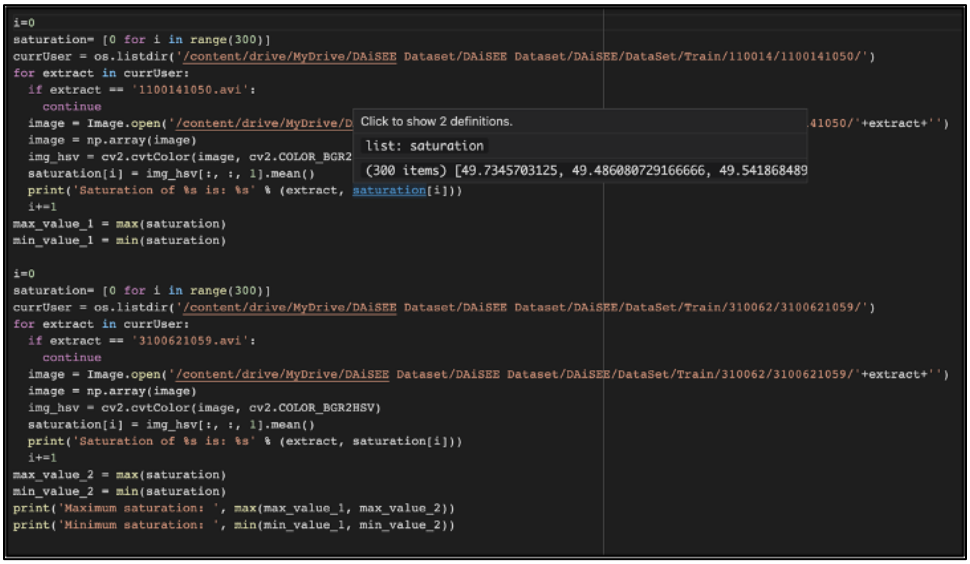
```
Brightness of 3100621059286.jpg is: 0.5066522216796886
Brightness of 3100621059287.jpg is: 0.5073001098632813
Brightness of 3100621059288.jpg is: 0.5066711807250979
Brightness of 3100621059289.jpg is: 0.506983858744304
Brightness of 3100621059290.jpg is: 0.5064443206787117
Brightness of 3100621059291.jpg is: 0.5067691421508788
Brightness of 3100621059292.jpg is: 0.5064464314778646
Brightness of 3100621059293.jpg is: 0.507161992390951
Brightness of 3100621059294.jpg is: 0.5070588302612307
Brightness of 3100621059295.jpg is: 0.5073498916625974
Brightness of 3100621059296.jpg is: 0.5069002024332682
Brightness of 3100621059297.jpg is: 0.5071874745686844
Brightness of 3100621059298.jpg is: 0.506918284098307
Brightness of 3100621059299.jpg is: 0.5076055145263679
Brightness of 3100621059300.jpg is: 0.5071639887491854
Maximum brightness: 0.7819975953407574
Minimum brightness: 0.42639817062858415
```

Note. Samples of brightness levels across image frames

Like brightness, the saturation of images also plays a huge role in how image frames are recognized. We compare the images to see how varied the saturation is using the script in Figure 22. There is only around 2% difference between the maximum and minimum saturation as shown in Figure 23 and hence no normalization of saturation is required.

Figure 22

Samples of saturation levels across image frames



Note. Exploration of saturation levels across image frames

Figure 23

Samples of saturation levels across image frames

```
Saturation of 3100621059284.jpg is: 48.77061197916667
Saturation of 3100621059285.jpg is: 48.79483072916667
Saturation of 3100621059286.jpg is: 48.740045572916664
Saturation of 3100621059287.jpg is: 48.67159830729167
Saturation of 3100621059288.jpg is: 48.648014322916666
Saturation of 3100621059289.jpg is: 48.73847981770833
Saturation of 3100621059290.jpg is: 48.795315755208335
Saturation of 3100621059291.jpg is: 48.832698567708334
Saturation of 3100621059292.jpg is: 48.576848958333336
Saturation of 3100621059293.jpg is: 48.559033203125
Saturation of 3100621059294.jpg is: 48.533147786458336
Saturation of 3100621059295.jpg is: 48.662978515625
Saturation of 3100621059296.jpg is: 48.5462890625
Saturation of 3100621059297.jpg is: 48.570201822916665
Saturation of 3100621059298.jpg is: 48.38095052083333
Saturation of 3100621059299.jpg is: 48.382718098958335
Saturation of 3100621059300.jpg is: 48.383369140625
Maximum saturation:  49.7345703125
Minimum saturation: 47.75248046875
```

Note. Samples of saturation levels across image frames

It is expected that different videos have different levels of resolution and that few image frames are clear, while the others are relatively blurred. We check the blurriness level of each image frame to decide if normalization of blur is required as shown in Figure 24. Here, we observe that around 22% of images are blurred, as seen from Figure 25. Since that is a significant number of images, we will be normalizing the same as a part of transformation.

Figure 24

Samples of blurriness levels across image frames

```
var= [0 for i in range(300)]
currUser = os.listdir('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/110014/1100141050/')
for extract in currUser:
  if extract == '1100141050.avi':
    continue
  image = Image.open('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/110014/1100141050/'+extract+'')
  image = np.array(image)
  grey = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  var[i] = cv2.Laplacian(grey, cv2.CV_64F).var()
  if var[i] < 120:
    blur_count+=1
  else:
    non_blur_count+=1
  i+=1
max_value_1 = max(var)
min_value_1 = min(var)
currUser = os.listdir('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/310062/3100621059/')
for extract in currUser:
  if extract == '3100621059.avi':
    continue
  image = Image.open('/content/drive/MyDrive/DAiSEE Dataset/DAiSEE Dataset/DAiSEE/DataSet/Train/310062/3100621059/'+extract+'')
  image = np.array(image)
  grey = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  var[i] = cv2.Laplacian(grey, cv2.CV_64F).var()
  if var[i] < 120:
    blur_count+=1
  else:
    non_blur_count+=1
  i+=1
max_value_2 = max(var)
min_value_2 = min(var)
print('Maximum Blur: ', max(max_value_1, max_value_2))
print('Minimum Blur: ', min(min_value_1, min_value_2))
print(f'Total number of non-blurred images = {non_blur_count}')
```

System Preferences

Note. Exploration of blurriness levels across image frames

Figure 25

Samples of blurriness levels across image frames

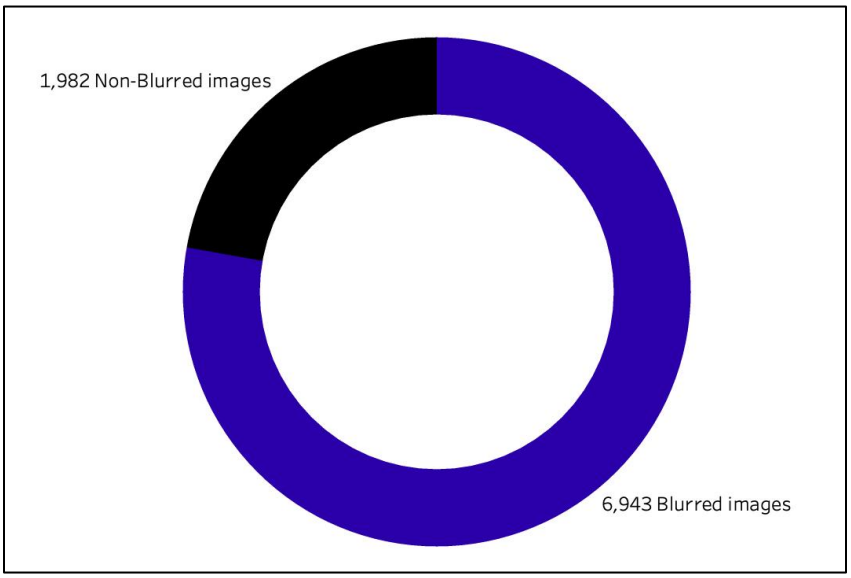
```
Blurriness index of 3100621059286.jpg = 88.89624969376457
Blurriness index of 3100621059287.jpg = 89.27772093874613
Blurriness index of 3100621059288.jpg = 89.5073879034678
Blurriness index of 3100621059289.jpg = 92.27048534948561
Blurriness index of 3100621059290.jpg = 90.24428151342603
Blurriness index of 3100621059291.jpg = 89.1034221780989
Blurriness index of 3100621059292.jpg = 88.75077934502495
Blurriness index of 3100621059293.jpg = 90.61293403303357
Blurriness index of 3100621059294.jpg = 90.08726057984036
Blurriness index of 3100621059295.jpg = 88.88821178511515
Blurriness index of 3100621059296.jpg = 88.87024606793722
Blurriness index of 3100621059297.jpg = 89.72264504156114
Blurriness index of 3100621059298.jpg = 89.98612853239909
Blurriness index of 3100621059299.jpg = 89.32267902627521
Blurriness index of 3100621059300.jpg = 89.85240231509738
Maximum Blur:  128.00851499413383
Minimum Blur:  75.96588539971248
Total number of non-blurred images = 6943
Total number of blurred images = 1982
```

Note. Samples of blurriness levels across image frames

As a part of the explorative analysis, we see from Figure 26 that around 22% of the image frames were blurred, allowing a threshold of 0.8 for blurriness. Since our dataset is humongous with over a million initial images, we used a subset of the training data for this analysis. 10% of the images were randomly chosen and checked for blurriness. The 22% of images being significantly blurred implies that the images could be from moments where the primary subject was moving, and hence might lead to inducing noise to the model. Images within the 80% blurriness threshold are normalized for the model to effectively predict irrespective of the image resolution.

Figure 26

Split of Blurred and Non-Blurred images



Note. Split of Blurred and Non-Blurred images in the dataset

3.4 Data Transformation

Data transformation is the process of converting data into different formats or structures based on requirements to make the target dataset well organized. Multiple expensive and open-

source ETL and ELT tools are used at enterprise level to achieve desired results. Image Transformation is performed during image processing to transform an image from one dimension to another. Viewing an image in dimensions leads to the discovery of features that may be challenging to see in image space. In this project we are implementing image smoothening, image normalization, image regularization techniques – Image resizing, grey scaling, horizontal flipping, rotation, affine and Singular value decomposition techniques for image transformation.

Image smoothing aims to remove unimportant fine-scale details while preserving main image structures. It is used to eliminate noise or generate a less pixelated picture. Most smoothing techniques rely on low-pass filtration. Another way is by calculating the average value of a set of image pixels. There are four different methods provided by the OpenCV. We are using the averaging method to blur the images. This will help remove noise related to high frequencies at the edges. Cv2.blur method takes the average of all pixels in the kernel region and substitutes it with the core feature as shown in Figure 27. Figure 28 shows the results of image smoothing. The total number of images smoothing are 111,026 in training data and 34,840 in validation data.

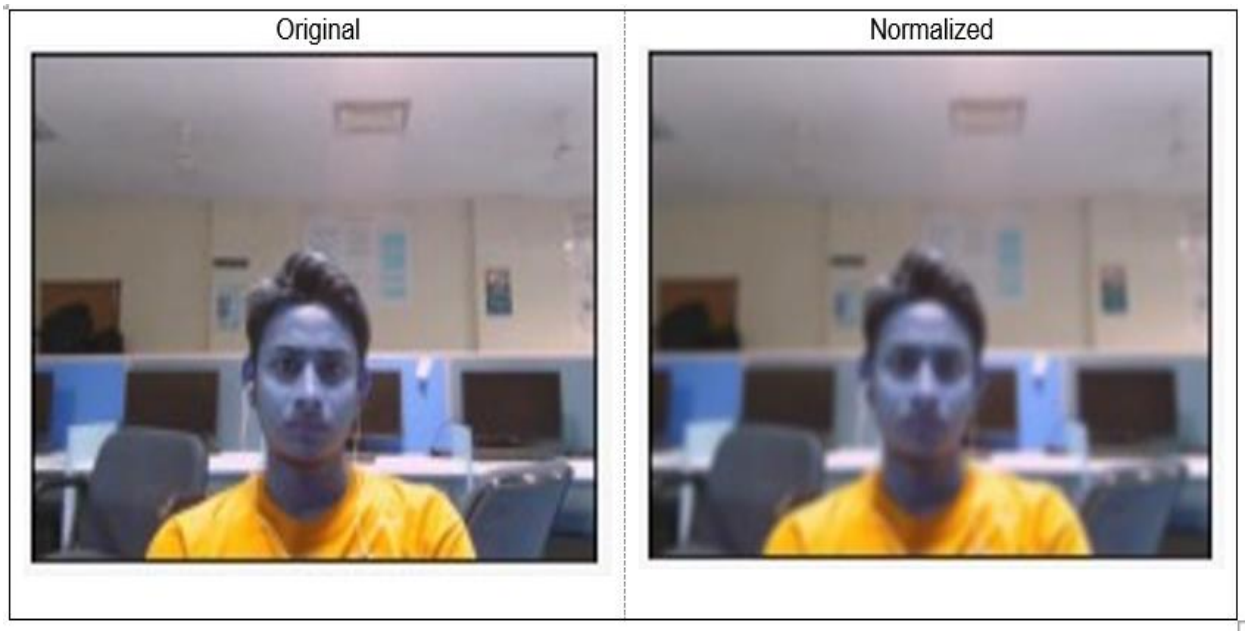
Figure 27

Image Smoothening Sample Code Snippet using cv2.blur method

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('31006210592.jpg')

blur = cv2.blur(img,(10,10))

plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

Figure 28

Image Smoothening Sample Image



Note. This figure illustrates before and after image smoothening results

Image normalization is used to adjust and normalize the contrast of images to make them friendly to the normal senses as videos can be captured by cameras of different resolutions. Normalize function of OpenCV library is used to perform the task and provide resultant image pixel range in 0-255. If any image has low contrast, it will increase and make it more defined. It

is also called contrast stretching. Figure 29 provides information of code implementation and Figure 30 is the sample of results achieved after processing. The total number of images normalized are 111,026 in training data and 34,840 in validation data.

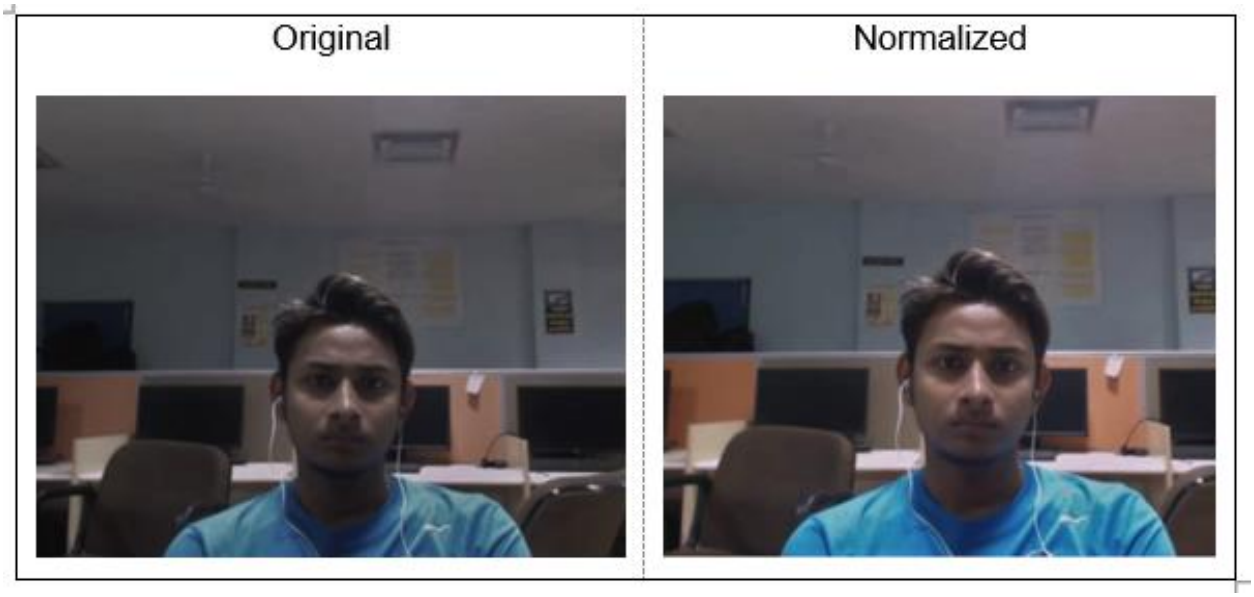
Figure 29

Image Normalization Sample Code Snippet using cv.normalize method

```
import cv2 as cv
import numpy as np
from google.colab.patches import cv2_imshow

img = cv.imread('31006210592.jpg')
norm_img = np.zeros((800,800))
final_img = cv.normalize(img,  norm_img, 0, 255, cv.NORM_MINMAX)
cv2_imshow( final_img)
cv.imwrite('city_normalized.jpg', final_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Figure 30

Image Normalization Sample Image.



Note. This figure illustrates before and after image normalization results

Regularization is used to avoid overfitting and to help the model generalize better. We can also reduce the overfitting by increasing the dataset. In the project videos are converted into image frames. To increase and diversify this image dataset we have implemented image rotation, scaling, shifting, shearing the range (shifting the image along the x and y-axis), cropping, resizing the image, and flipping the image. This technique of diversifying the existing image data without adding new images is called data augmentation. Data transformation of image frames is performed using python libraries PIL, PyTorch, and torchvision. (Jain, 2018)

Image Sizing is implemented because videos are collected from various sources captured via cameras with different resolutions. It is the process of standardizing all images to one standard size to maintain consistency of image composition. Resize function of OpenCV library is used where image dimensions are set to (height=300, width=300) and INTER_AREA interpolation used to resample the images using relationship of pixel area. Figure 31 provides information of code implementation and Figure 32 is the sample of results achieved after processing. The total number of images resized are 111,026 in training data and 34,840 in validation data.

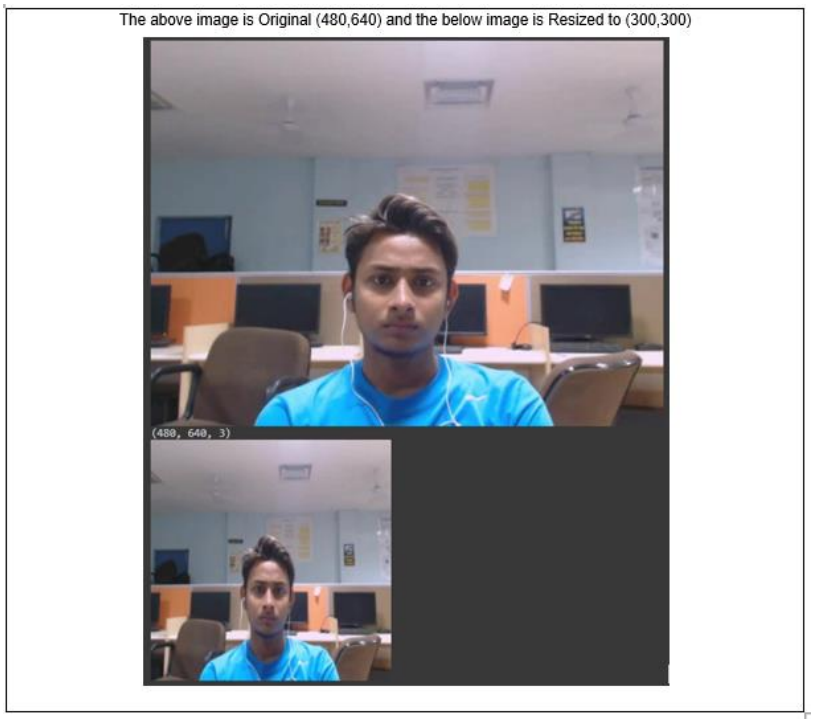
Figure 31

Image Resizing Sample Code Snippet

```
[17] image=cv2.imread('31006210592.jpg')
     cv2_imshow(image)
     print(image.shape)
     h1=300
     w1=300
     dimension = (w1, h1)
     resized_image = cv2.resize(image, dimension, interpolation = cv2.INTER_AREA)
     cv2_imshow(resized_image)
```

Figure 32

Image Resize Sample Image

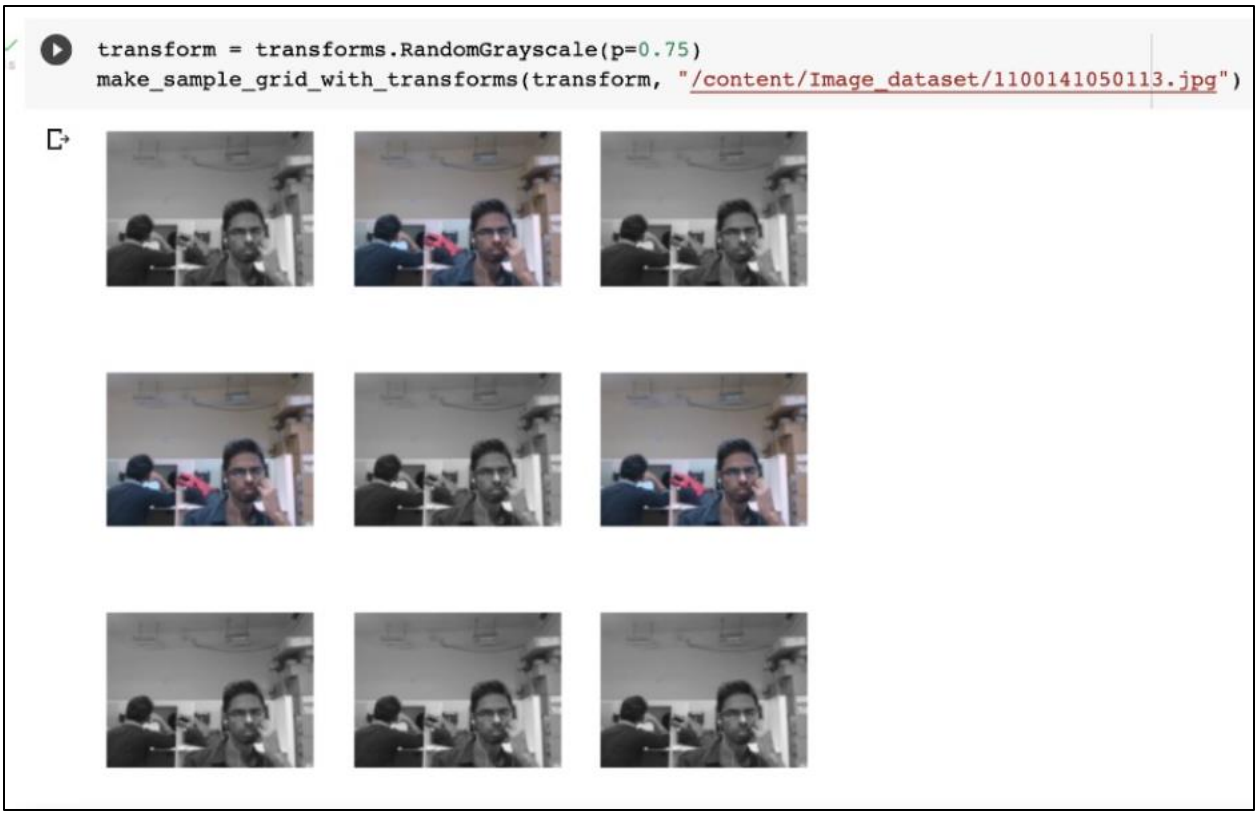


Note. This figure illustrates before and after image resize results

Gray scaling of the image is performed to simplify code complexity and reduce computational cost as the images are huge in number. The project utilizes Torchvision library of python and RandomGreyscale method with probability set to p = 0.75. This means the probability of image grey scaling is 0.75. Figure 33 shows the code snippet used for implementation with samples of resultant images after grey scaling. 50% of the images are used for grey scaling for each training and validation dataset.

Figure 33

Image Grey Scaling

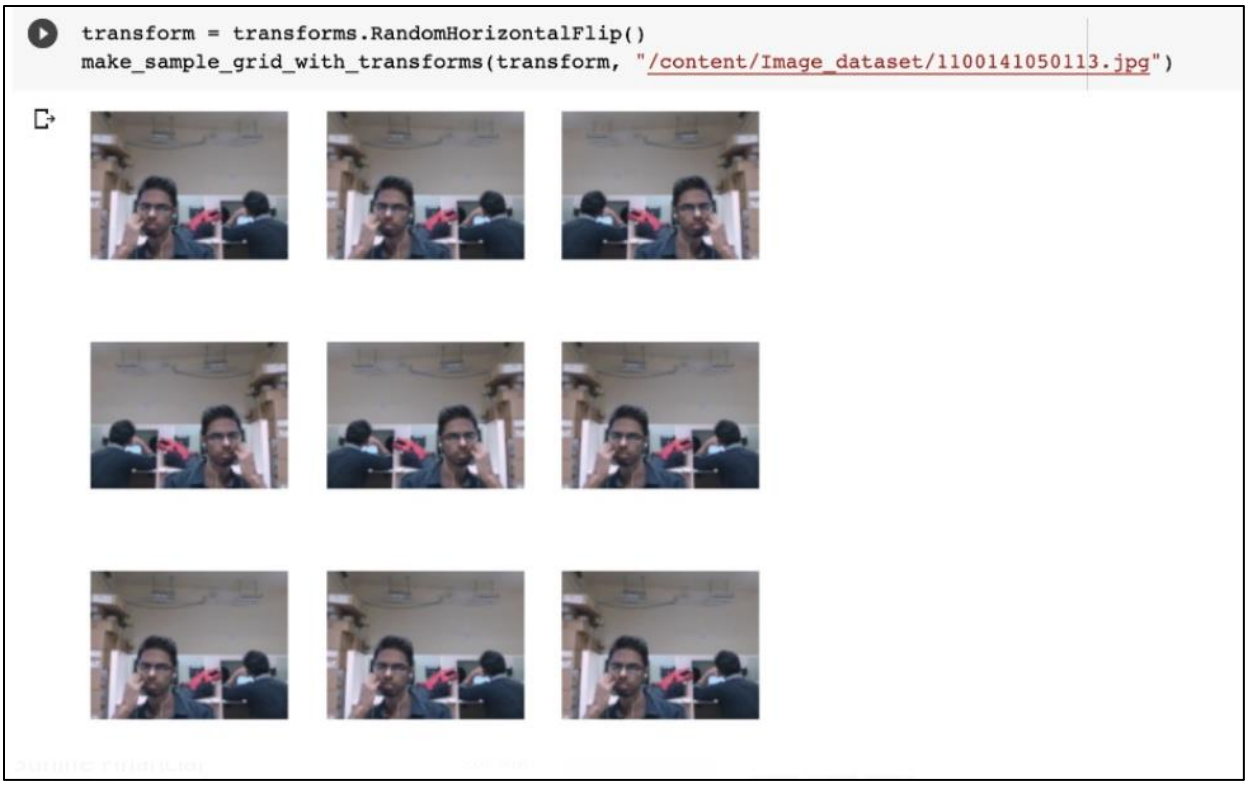


Note. This image illustrates sample of results of grey scaling implementation

Horizontal flipping helps in training our model and adding more features by flipping video images over a horizontal axis, this, in turn, makes the model competent with different cameras. As part of the project, we use Torchvision library RandomHorizontalFlip function to transform the images. The probability is by default 0.5. Figure 34 shows the code snippet used for implementation with samples of resultant images after grey scaling. Five percent of original image frames are used for horizontal flipping and the total number of images generated is 44,410 for training data and 13,936 for validation data.

Figure 34

Image Horizontal Flipping.

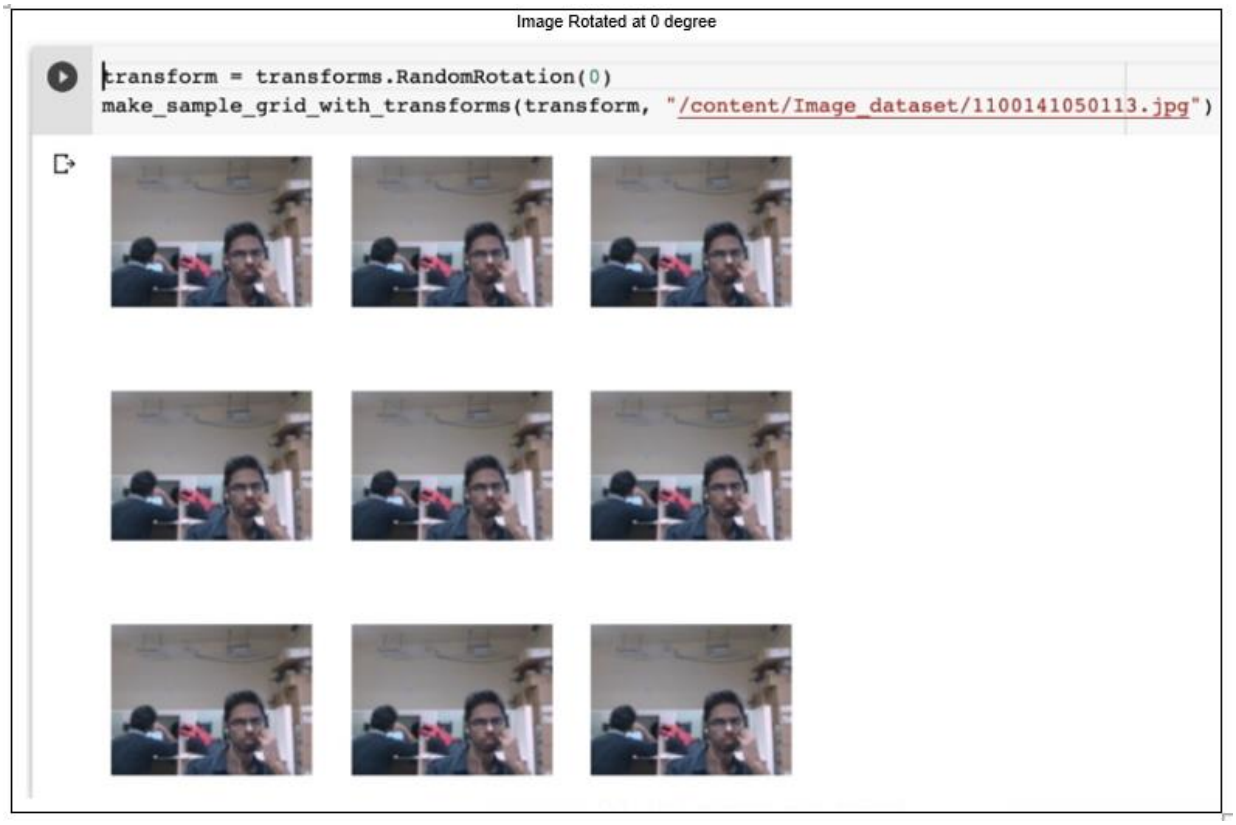


Note. This image illustrates sample of results of horizontal flipping implementation

Image rotation is necessary as the position of participants in the video will not be consistent. As part of the project images are rotated on degrees of 0-20-45-90 to train the model on multiple positions and angles. This will reduce the chances of overfitting of models. The Torchvision library's RandomRotate function is used to transform the images with different degree levels. Figure 35-38 shows the code snippet used for implementation with samples of resultant images after image rotation across different degrees. One percent of input image frames are used for each image rotation degree and the resultant images are multiplied by 8 for each degree of rotation. The total resultant images generated are 26,646 for training data and 8,361 for validation data.

Figure 35

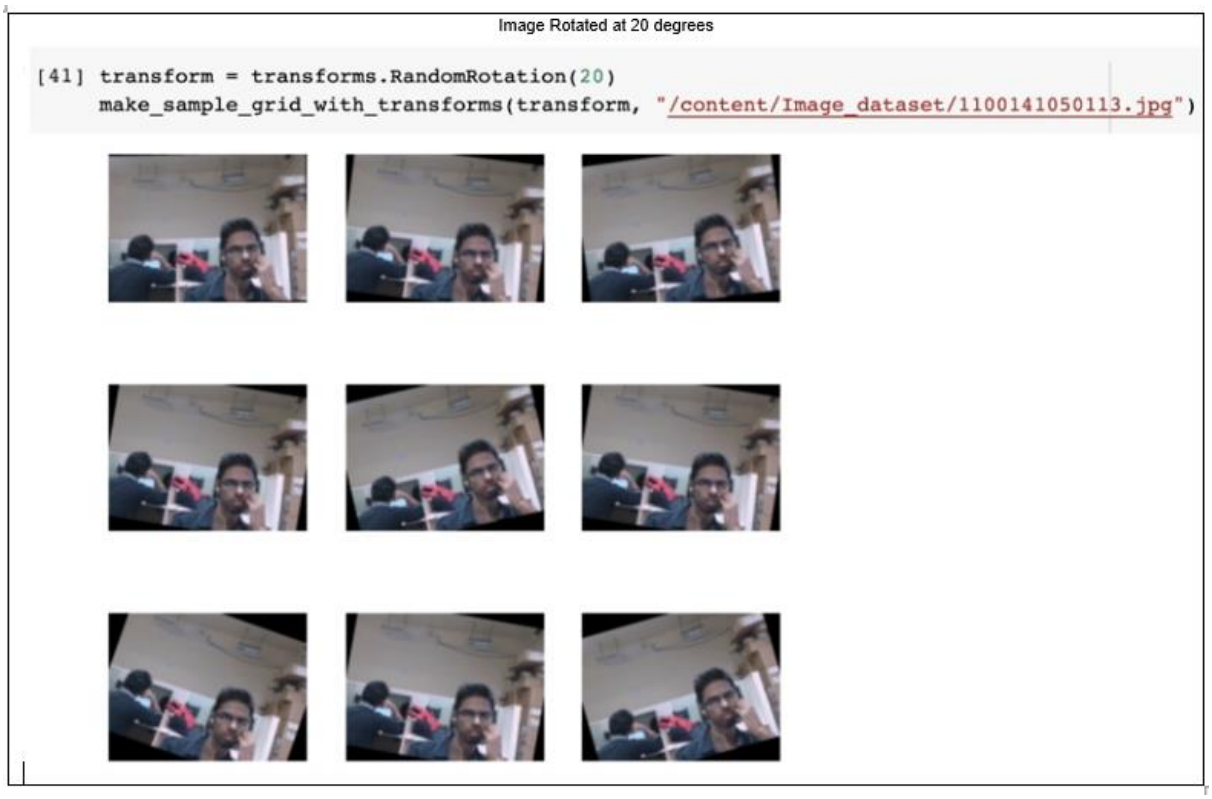
Image Rotation at 0 degree



Note. This image illustrates sample of results of image rotation at 0-degree implementation

Figure 36

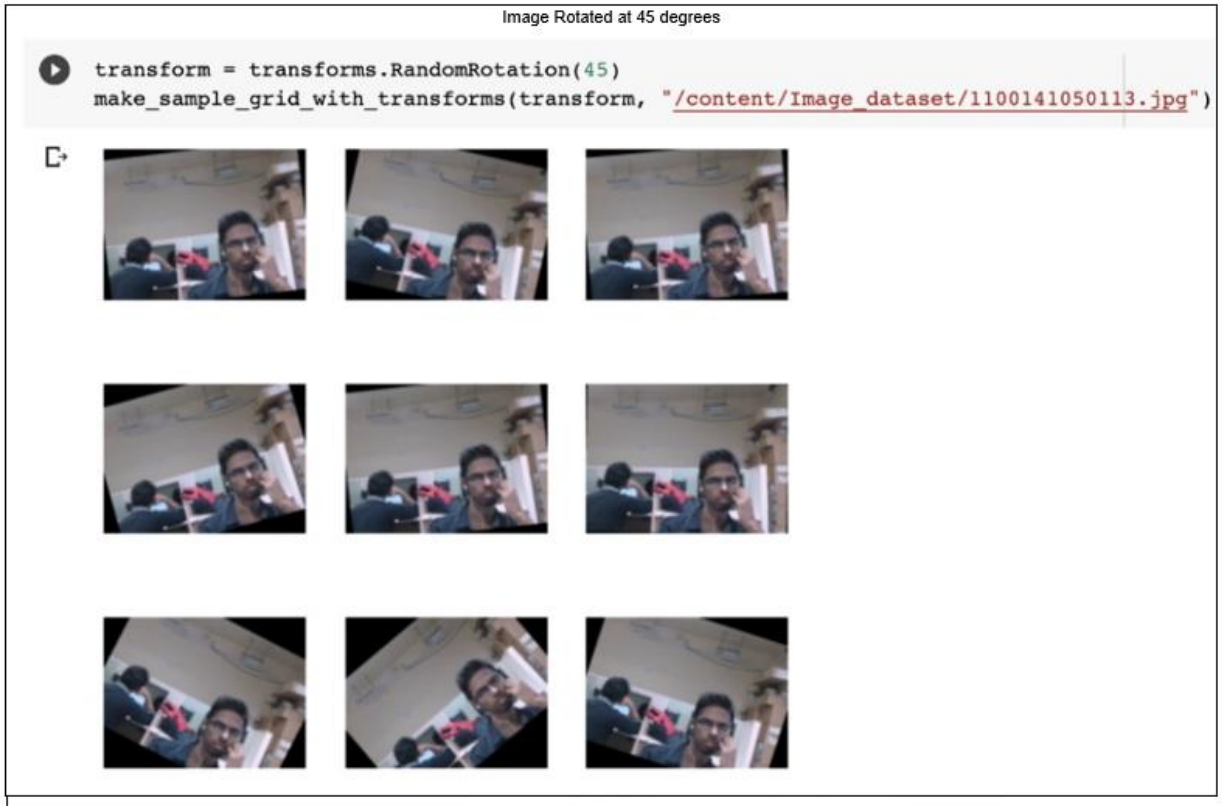
Image Rotation at twenty degrees



Note. This image illustrates sample of results of image rotation at twenty degrees implementation

Figure 37

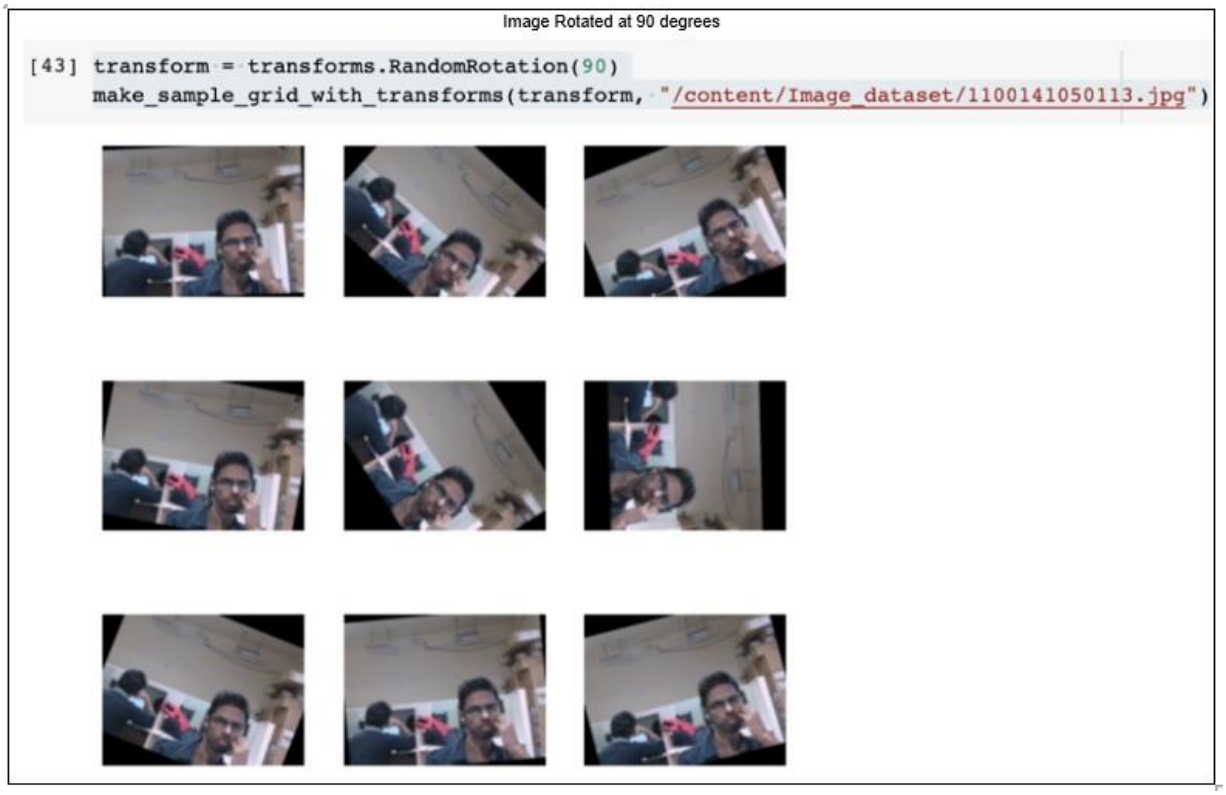
Image Rotation at forty-five degrees.



Note. This image illustrates sample of results of image rotation at forty-five degrees implementation

Figure 38

Image Rotation at ninety degrees

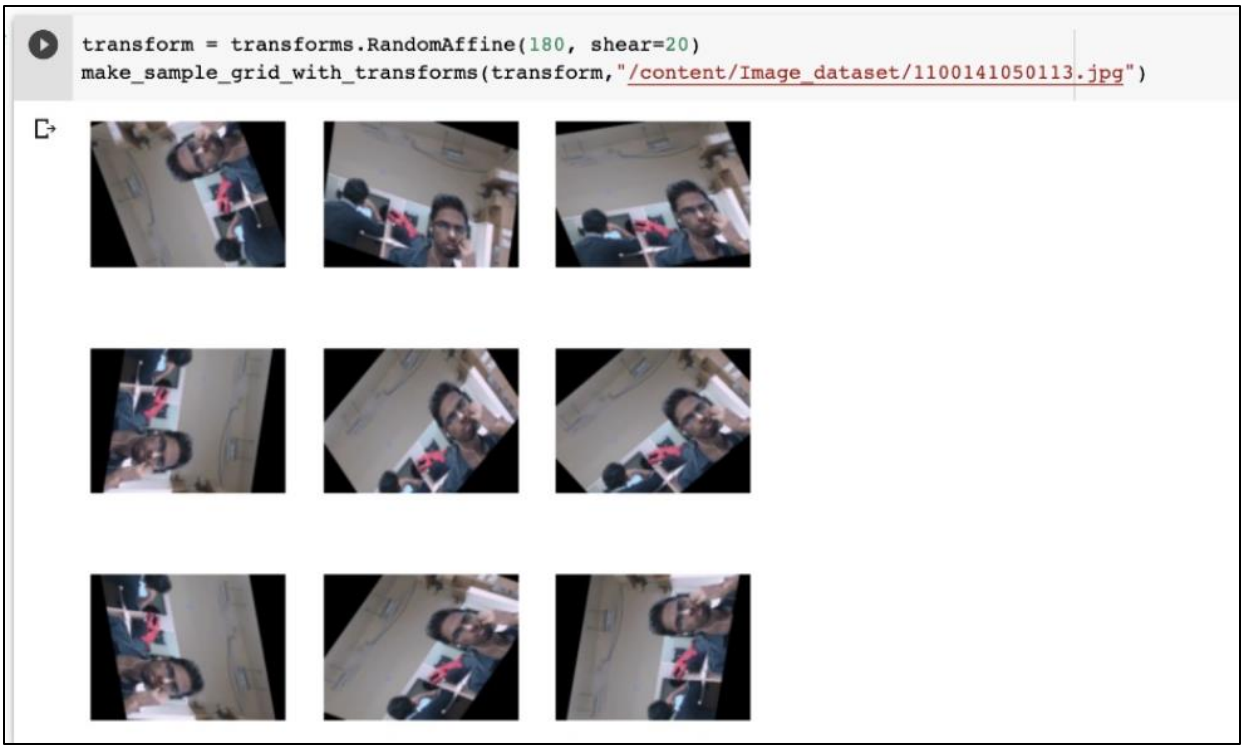


Note. This image illustrates sample of results of image rotation at ninety degrees implementation

Random affine is used to correct spatial biases and deflection caused by non-optimal camera angles without varying the center. This helps in changing shape and size along the axis in two-dimensional space. Torchvision library's RandomAffine method is adapted for implementation where shearing is set at degree of -20 and +20 parallel to horizonal axis. Five percent of original image frames are utilized for this transformation and the resultant images are multiplied by 8. The total number of images generated is 44,410 for training data and 13,936 for validation data. Figure 39 shows implementation of Random affine.

Figure 39

Image Random Affine.



Note. This image illustrates sample of results of Random Affine implementation

Dimensionality reduction is done to reduce the number of dimensions or columns present in a dataset. It transforms the dataset to have less dimensions while still containing most information. In image dataset, dimensionality reduction is done by compressing the image size but having the utmost quality possible. It is done by using the matrix formula $X \cong U \sum V \, 〚^{\wedge}T〛$ where in it takes just the approximation of few columns' times few rows. Singular value decomposition is a method that reduces a complex matrix into smaller portions to simplify computation of succeeding matrices. By using SVD we do not need to store all rows and all columns since it approximates and gives us the result. The implementation focuses on the image and return 3 matrices, each corresponding to one channel (R, G and B channels), compress each channel matrix separately using NumPy python library's linear algebra SVD function and then merging each single compresses channel into resultant image. The resultant image is compressed and there is a shift in the file size from 786432 Kb to 492000Kb. Thus, SVD has compressed the

image to 62.56% of its original size. Figure 40 is a code snippet of the function used to compress single channel using SVD. Figure 41 provides statistics of results after the implementation of SVD on a sample image.

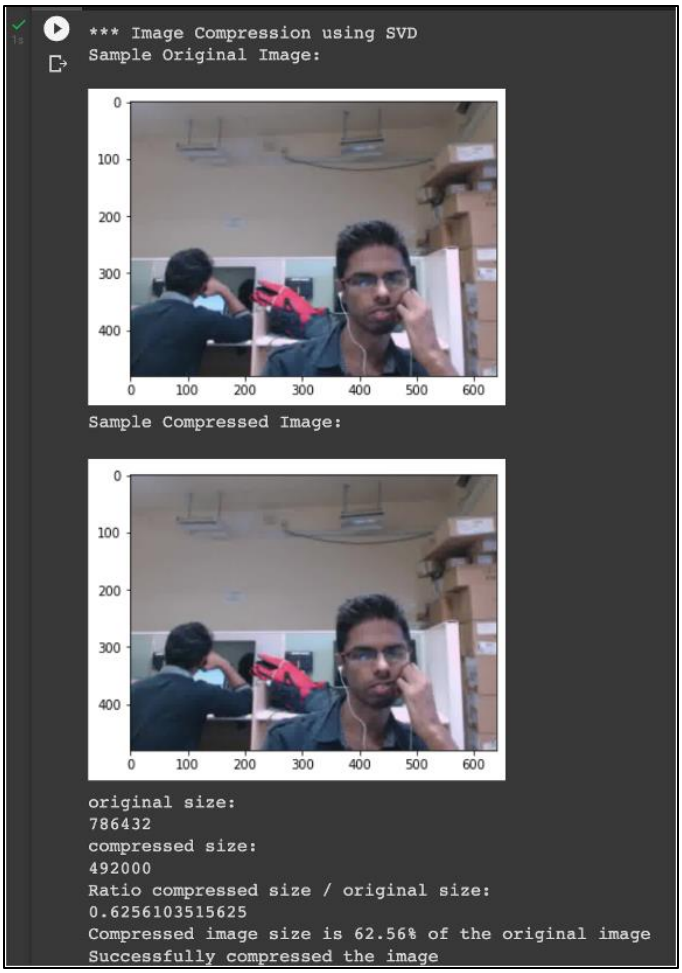
Figure 40

Image compression SVD function code snippet

```
# compress the matrix of a single channel
def compressSingleChannel(channelDataMatrix, singularValuesLimit):
    uChannel, sChannel, vhChannel = numpy.linalg.svd(channelDataMatrix)
    aChannelCompressed = numpy.zeros((channelDataMatrix.shape[0], channelDataMatrix.shape[1]))
    k = singularValuesLimit

    leftSide = numpy.matmul(uChannel[:, 0:k], numpy.diag(sChannel)[0:k, 0:k])
    aChannelCompressedInner = numpy.matmul(leftSide, vhChannel[0:k, :])
    aChannelCompressed = aChannelCompressedInner.astype('uint8')
    return aChannelCompressed
```

Figure 41

Image Compression using SVD Statistics



Note. This image illustrates sample of results of image compression using SVD

3.5. Data Preparation

Once data preprocessing and transformation is completed, we will need to prepare the data to make it ready for modeling. The image frames will need to be split into training, validation and test datasets to train and evaluate the models. The split should be done ensuring the data distribution is the same in all the three sets, so that the models get unbiased and true sets of data to train on. The DAiSEE dataset already has train, test and validation data available in separate folders. For the model training and evaluation however, we need to create our own test dataset from the training dataset, as the available test dataset is not labeled. The validation dataset available is labeled and can be used directly for validation. For splitting the training

dataset into train and test, we are using 80-20 split, with 80% of the data retained for training and 20% for testing the models. We have used NumPy's split function after shuffling the dataset for the split. After the train-test split, we will have 236,244 image frames for training, 59,061 image frames for validation and 37,740 image frames for testing, samples of which are shown in Figure 42-Figure 44.

Figure 42

Splitting the data into train and test

```
test_ratio=0.8
allFileNames = os.listdir('/content/drive/MyDrive/DAiSEE/DAiSEE/DataSet/Train')
np.random.shuffle(allFileNames)
train_FileNames, test_FileNames = np.split(np.array(allFileNames),[int(len(allFileNames)* (1 - test_ratio))])
```

Note. Splitting the data for training and testing with 80-20 ratio

Figure 43

Number of images after train-test split

```
Test dataset: Total number of image frames: 333117

Train dataset: Total number of image frames: 1332469

Validation dataset:Total number of image frames: 522600
```

Note. Number of images after train-test split

Figure 44

Samples of Train, Test and Validation Data

Test Dataset:

Out[10]:

	ImageID	Boredom	Engagement	Confusion	Frustration
0	5000441001.jpg	1	2	0	0
1	5000441002.jpg	0	2	0	0
2	5000441003.jpg	1	2	0	0
3	5000441005.jpg	2	2	0	0
4	5000441006.jpg	2	2	1	2
...	...	...	...	...	...
1779	987736029.jpg	0	3	0	0
1780	998826011.jpg	0	3	0	0
1781	9988260112.jpg	0	3	0	0
1782	9988260115.jpg	0	3	0	0
1783	9988260119.jpg	1	2	0	0

37740 rows x 5 columns

Train Dataset:

Out[11]:

	ImageID	Boredom	Engagement	Confusion	Frustration
0	1100011002.jpg	0	2	0	0
1	1100011003.jpg	0	2	0	0
2	1100011004.jpg	0	3	0	0
3	1100011005.jpg	0	3	0	0
4	1100011006.jpg	0	3	0	0
...	...	...	...	...	...
5353	4599990246.jpg	0	3	0	0
5354	4599990247.jpg	0	3	0	0
5355	4599990248.jpg	1	2	1	1
5356	4599990249.jpg	0	3	0	0
5357	459999025.jpg	1	3	0	0

236244 rows x 5 columns

Validation Dataset:

Out[12]:

	ClipID	Boredom	Engagement	Confusion	Frustration
0	4000221001.avi	0	2	0	0
1	4000221002.avi	1	3	0	0
2	4000221006.avi	1	2	0	0
3	4000221008.avi	0	3	0	0
4	4000221009.avi	2	2	0	0
...	...	...	...	...	...
1424	5912920240.avi	1	3	0	1
1425	5912920241.avi	3	3	0	0
1426	5912920242.avi	1	2	0	0
1427	5912920243.avi	1	2	1	0
1428	5912920244.avi	1	2	1	1

59061 rows x 5 columns

Note. Samples of Train, Test and Validation Data

3.6. Data Statistics

As a part of the Data Engineering process, we have collected raw data, performed data cleaning, exploratory data analysis, transformations, and data preparation steps to make the data ready for modeling summary of which is detailed in table 3.

Table 3

Summary of data processing techniques

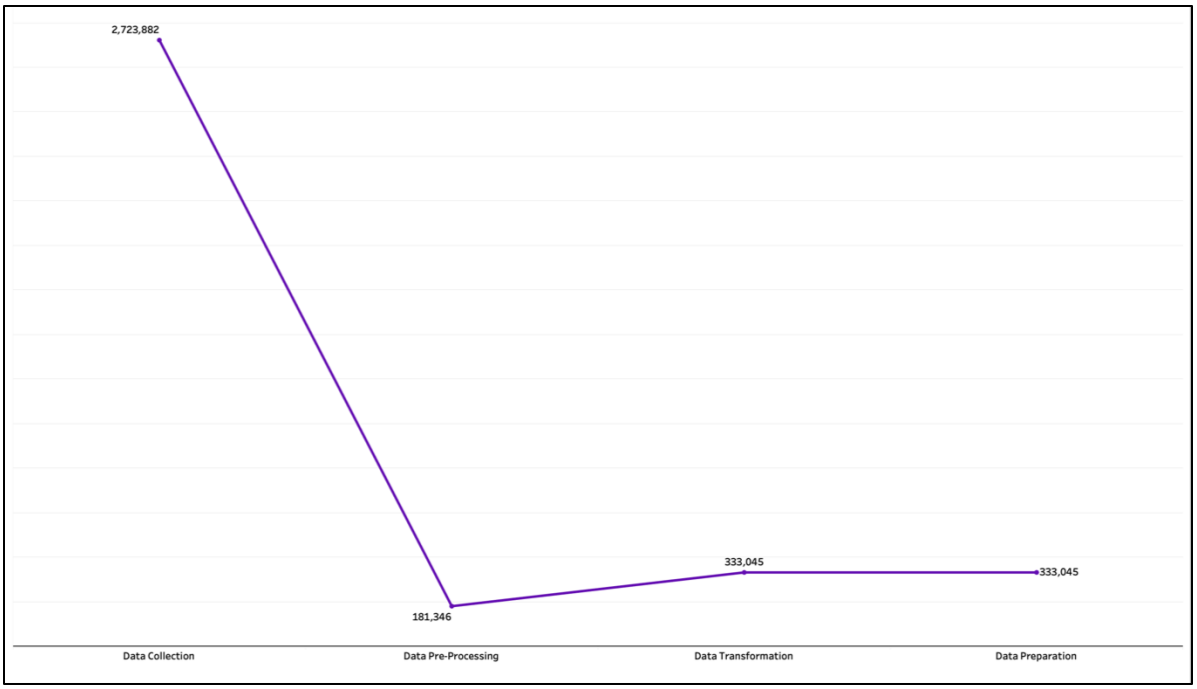
Process	Method	Count
Data Collection	Raw Data	9068 videos
Data Collection	Conversion to image frames : Total 9068 videos were converted to image frames	2723882 images
Data PreProcessing	Removing Duplicates : We removed duplicate image frames with 85% as similarity threshold	190428 images
Data PreProcessing	Removing Missing Data : We removed missing data i.e image where no student was present	190416 images
Data PreProcessing	Handling Noisy Data : We Identifyied and removed blurred images with a threshold of 0.8	181360 images
Data PreProcessing	Handling Inconsistent Data : Inconsistent data was manually removed	181346 images
Data Transformation	Image Smoothening : We have used averaging method to blur the images	181346 images
Data Transformation	Image Normalization : Normalize function was used that resulted in image from 0-255 pixel range	181346 images
Data Transformation	Resizing Images : To bring all the images to optimal dimension	181346 images
Data Transformation	Gray Scaling : Torchvision library of python and RandomGreyscale method was used with probability of 0.75	181346 images
Data Transformation	Horizontal Flipping : It is to train the model by adding more features and flipping image over horizontal axis	239692 images
Data Transformation	Image Rotation : Images are rotated on degrees of 0-20-45-90 to train the model	274699 images
Data Transformation	Random Affine : Torchvision library's RandomAffine method is used	333045 images
Data Transformation	SVD : It will approximate without storing all rows and columns and will yield the result	333045 images
Data Preparation	Train-Test Split : Will result in 236,244 images for training ; 59,061 images for validation and 37,740 image frames for testing	333045 images

Note. Summary of data processing techniques

A high-level view of how the data quantity changes in each step of data engineering can be found in Figure 45.

Figure 45

Quantity of data in each stage

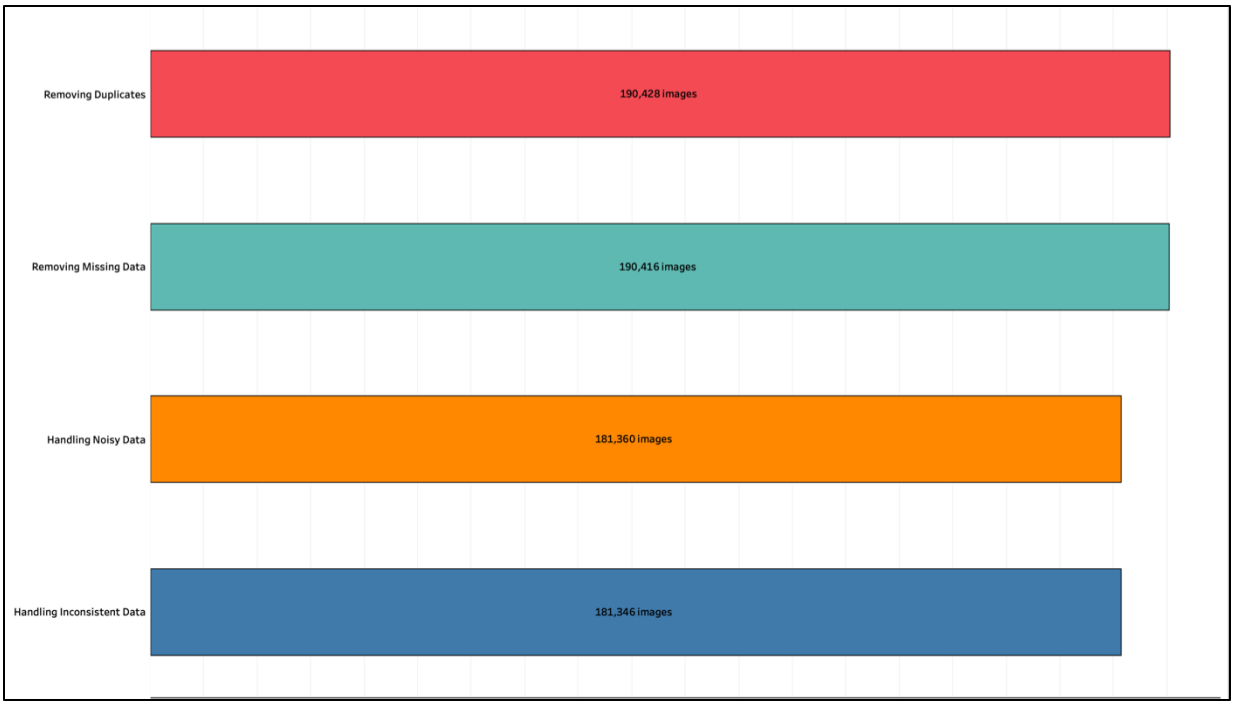


Note. Quantity of data in each stage from Data Collection to Data Preparation

A step-by-step variation in the number of image frames in the process of data pre-processing can be found in Figure 46.

Figure 46

Quantity of data in each stage of Data Pre-Processing

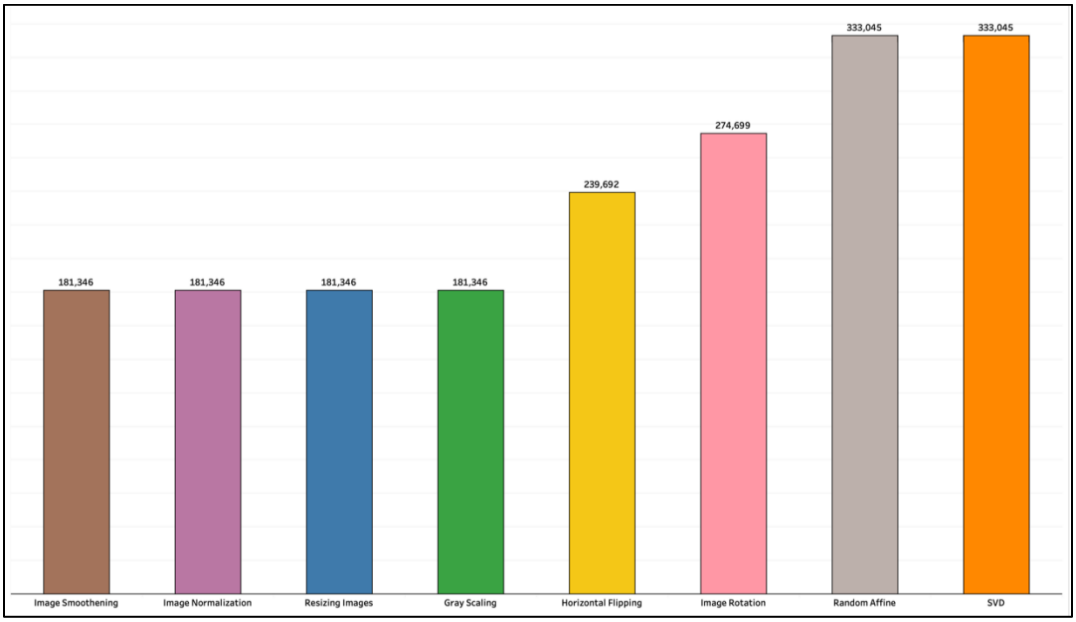


Note. Quantity of data in each stage from Data Pre-Processing

The variation of data volume through each step of Data Transformation performed can be found in Figure 47.

Figure 47

Quantity of data in each stage of Data Transformation

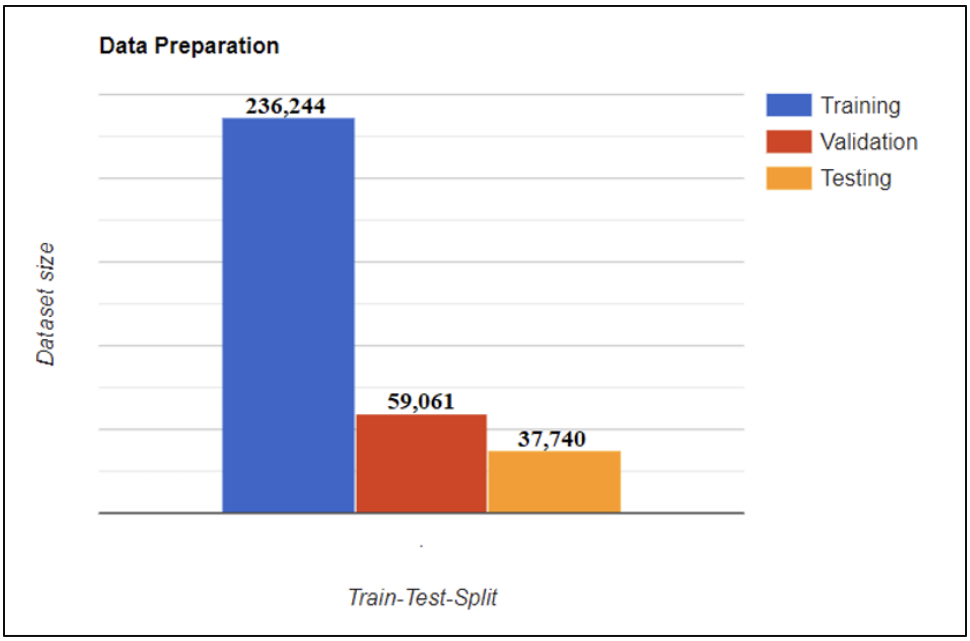


Note. Quantity of data in each stage from Data Transformation

The count of images in train, test and validation datasets after transformation is depicted in Figure 48.

Figure 48

Count of images in Train, Test and Validation Datasets



Note. Count of images in Train, Test and Validation Datasets

3.7. Data Analytics Results

The dataset we have consists of videos where participants portray one of the 4 expressions – Boredom, Engagement, Confusion and Frustration, each with intensities ranging from 0 to 3. A quick look at the basic statistics of the data can be seen in Figure 49.

Figure 49

Univariate Analysis

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8925 entries, 0 to 8924
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ClipID       8925 non-null   object
 1   Boredom      8925 non-null   int64
 2   Engagement   8925 non-null   int64
 3   Confusion    8925 non-null   int64
 4   Frustration  8925 non-null   int64
dtypes: int64(4), object(1)
memory usage: 348.8+ KB
```

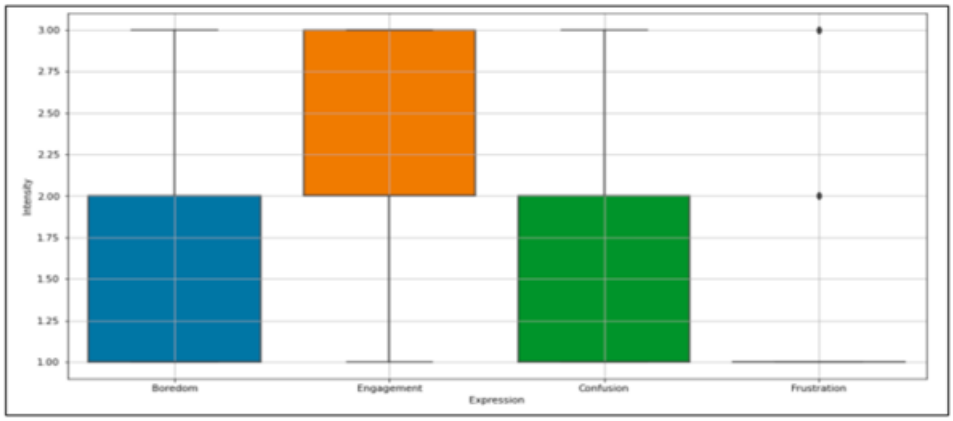
	Boredom	Engagement	Confusion	Frustration
count	8925.000000	8925.000000	8925.000000	8925.000000
mean	0.861176	2.382073	0.438655	0.285714
std	0.878869	0.615715	0.692508	0.581678
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	0.000000
50%	1.000000	2.000000	0.000000	0.000000
75%	2.000000	3.000000	1.000000	0.000000
max	3.000000	3.000000	3.000000	3.000000

Note. Univariate Analysis of Label information

The following box plots depict the distribution of these expressions and their intensities across the videos. We can observe that the data is imbalanced, with very few high intensity videos for frustration, and very few low intensity videos for engagement as depicted in Figure 50.

Figure 50

Box Plot depicting distribution of expressions and their intensities

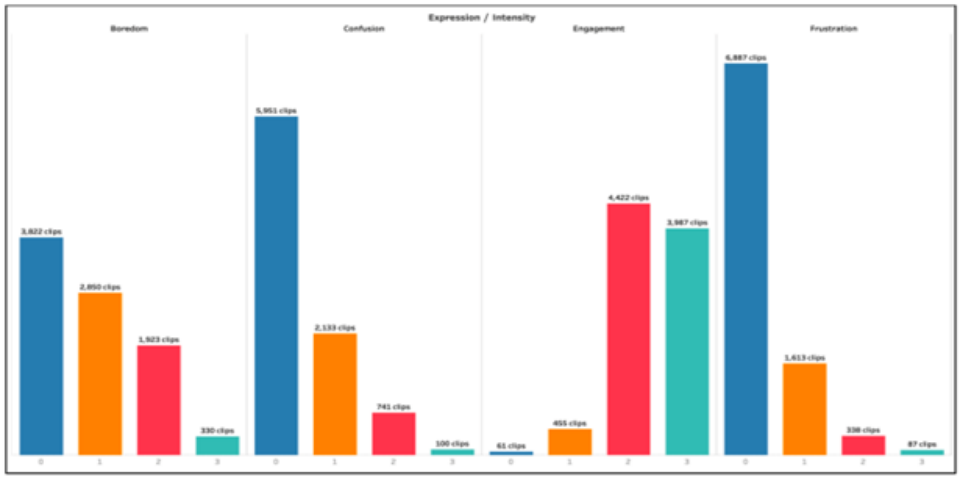


Note. Box Plot depicting distribution of expressions and their intensities

The total number of video clips for each emotion is the same, the distribution over different intensity levels vary for each expression as shown in Figure 51.

Figure 51

Histogram depicting number of video clips for each emotion



Note. Histogram depicting number of video clips for each emotion

The expressions portrayed by students need to be unique for each video. This means there should not be videos which have the same levels of intensity for more than one expression. We can deduce that from the correlation plot below in Figure 52. Since the levels of correlation of intensity is very low between different expressions, the videos are unique.

Figure 52

Correlation of intensities

	Boredom	Engagement	Confusion	Frustration
Boredom	1.000000	-0.416563	0.177947	0.249665
Engagement	-0.416563	1.000000	-0.131092	-0.205026
Confusion	0.177947	-0.131092	1.000000	0.407381
Frustration	0.249665	-0.205026	0.407381	1.000000

Note. Correlation of intensities between each engagement expression

4. Model Development

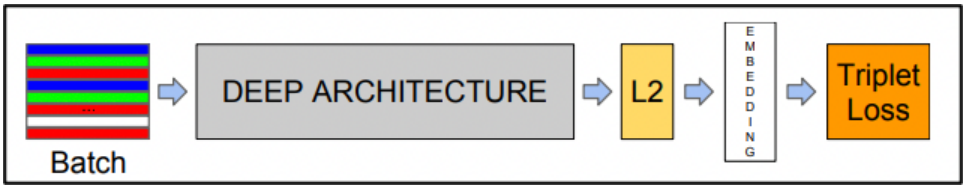
4.1. Model Proposals

Model development to identify the engagement of students can be split into two major components. The first component will be to extract the facial space and the essential features within it from the preprocessed data. The second part will be to use these facial features to develop a classification algorithm that can identify the engagement level.

The FaceNet algorithm extracts the face from an image and transforms the images into Euclidean space, where the distance between images directly correlates to their similarity. The uniqueness of the approach lies in the fact that the conversion to Gaussian space (also called embedding) happens on the result of CNN bult specifically to optimize the embedding. The optimization of this step is to identify a feature space where the mean squared error of distance between all faces, irrespective of the illumination, direction, etc. is the highest. This is in the interest of capturing even the tiniest of differences between faces. The formulation of this function is called triplet loss, which tries to cluster different images of the same face together, while creating a margin between different people's faces. Figure 53 represents the model architecture of FaceNet.

Figure 53

Architecture of FaceNet Algorithm



Note. Model architecture of FaceNet from "FaceNet: A Unified Embedding for Face Recognition and Clustering" by Schroff, F., Kalenichenko, D., & Philbin, J (2015), *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2015* (https://arxiv.org/abs/1503.03832)

The steps involved in the training process for finding the best triplet loss function are as follows:

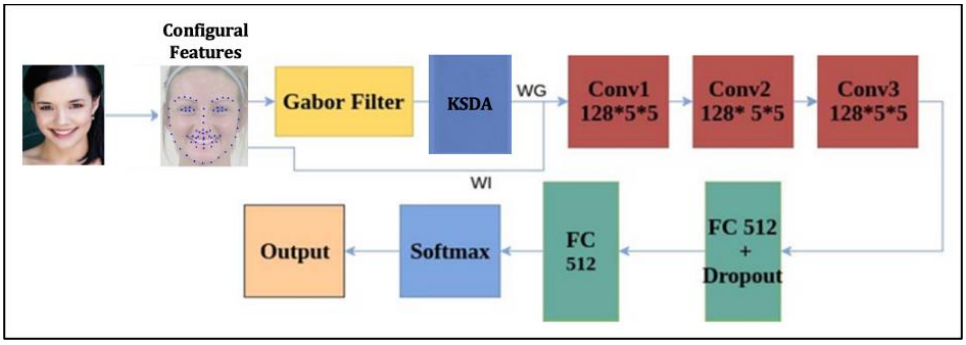
1. Choose an arbitrary image as the anchor image
2. Arbitrarily choose another image of the same person, to act as the positive image for the anchor image
3. Arbitrarily choose an image of another person to act as the negative image for the anchor image
4. Set the FaceNet loss function parameter α such that the anchor and positive image are closer together least and the anchor and negative image lie farther apart
5. Steps 1-4 are repeated by adding more and more images as anchor, positive and negative images, and optimizing α at each step

To perform the classification, we use the EmotioNet algorithm (Benitez-Quiroz et al., 2016). The EmotioNet algorithm first identifies the configural features in the face space. The configural features identified in the algorithm are pre-defined landmarks defined in (Neth & Martinez, 2009). Once the vector of configural features is obtained, they are normalized to make all images have the same inter-eye distance, whilst using midpoint between the corner of the eyes landmarks as the center of the eye. The algorithm then defines a feature vector the face space using a function of the Euclidean distances between the landmarks. On the facial feature vectors obtained, Gabor filters are used to detect the variation in the face due to skin deformation and

shadows. The configural feature vectors are then projected into a kernel space using Kernel Subclass Discriminant Analysis with RBF kernel to help identify the latent and non-linear aspects of the images. EmotioNet then feeds the kernel-projected feature vector as the input to a Residual Neural Network. The Residual Neural Network contains three 128*5*5 convolutional layers, two Fully Connected layers and a SoftMax layer at the end. The architecture of EmotioNet algorithm is shown in Figure 54. The output of the EmotioNet classifier is the result of the SoftMax layer with four probabilities, which represent the Action Unit intensity of each emotion. The probability values are mapped into intensity scale of 0 to 3 for each emotion.

Figure 54

Architecture of EmotioNet



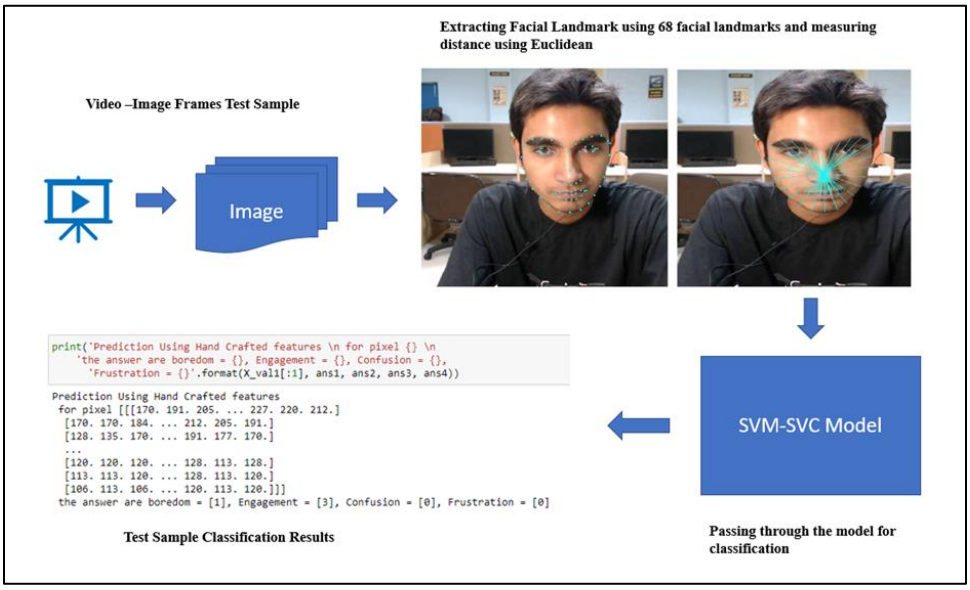
Note. Architecture of EmotioNet from "EmotioNet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild" by Benitez-Quiroz, C. F., Srinivasan, R., & Martinez, A. M (2016), *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (https://ieeexplore.ieee.org/abstract/document/7780969)

In modelling using SVM, the input data is in the form of videos which are converted into image frames. The model can predict the level of each class. It consists of two techniques; the first technique uses 68 facial landmark identification features to extract the features and make

predictions using SVM algorithm. Second, the feature extraction using custom CNN to extract the features and then pass these features to SVM algorithm to make predictions. The problem was solved using multi class classification concept, where each model is trained for each individual class, each model was able to predict each class with levels. The result of these models is concatenated to predict the overall classes with their levels. Figure 55 and Figure 56 provide an overview of the proposed models.

Figure 55

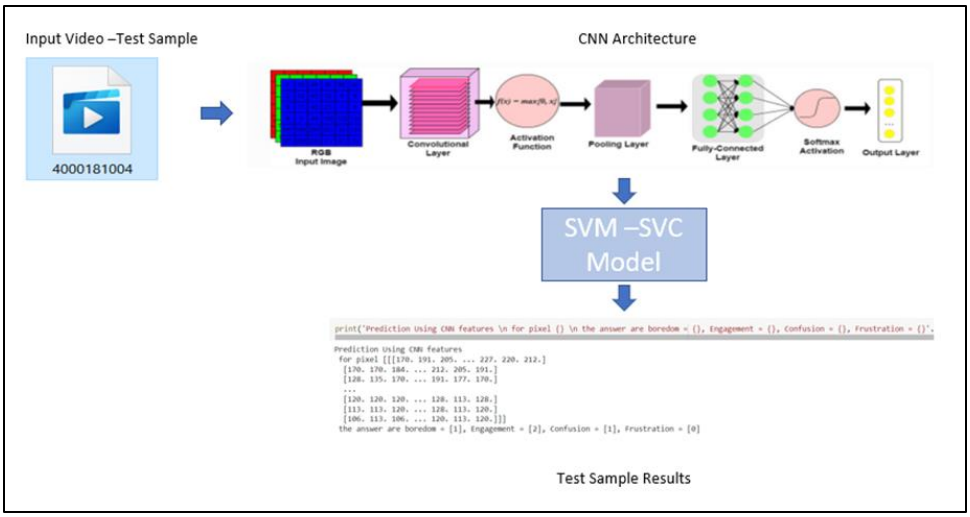
68 Facial Landmark Detection with SVM Process



Note. Overview of proposed model -Handcraft Feature Extraction +SVM-SVC

Figure 56

Custom CNN with SVM Process

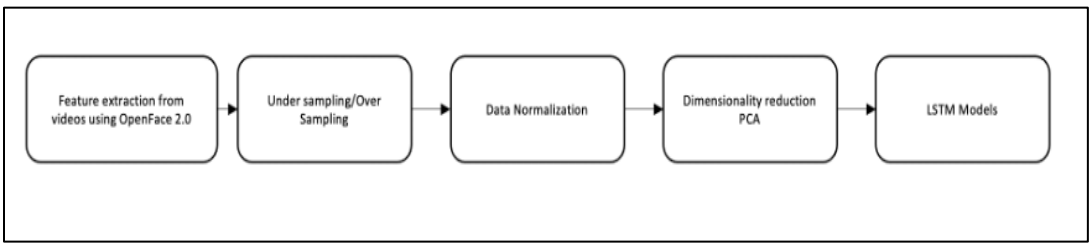


Note. Overview of proposed model custom CNN with SVM-SVC

In modelling using LSTMs. OpenFace is used for feature extraction followed by MTCNN as a face detector. CE-CLM which is a landmark detector resulted in 329 dimensions of features. Eye gaze, Head pose, and AU were extracted (Nur Karimah et al., 2021). Figure 57 shows the proposed architecture of LSTM model.

Figure 57

Proposed Model Architecture



Note. Proposed Model Architecture

OpenFace was used to extract features such as Head Pose, AU and gazing of eye. We then did undersampling and oversampling followed by normalization and PCA. The refined data is then

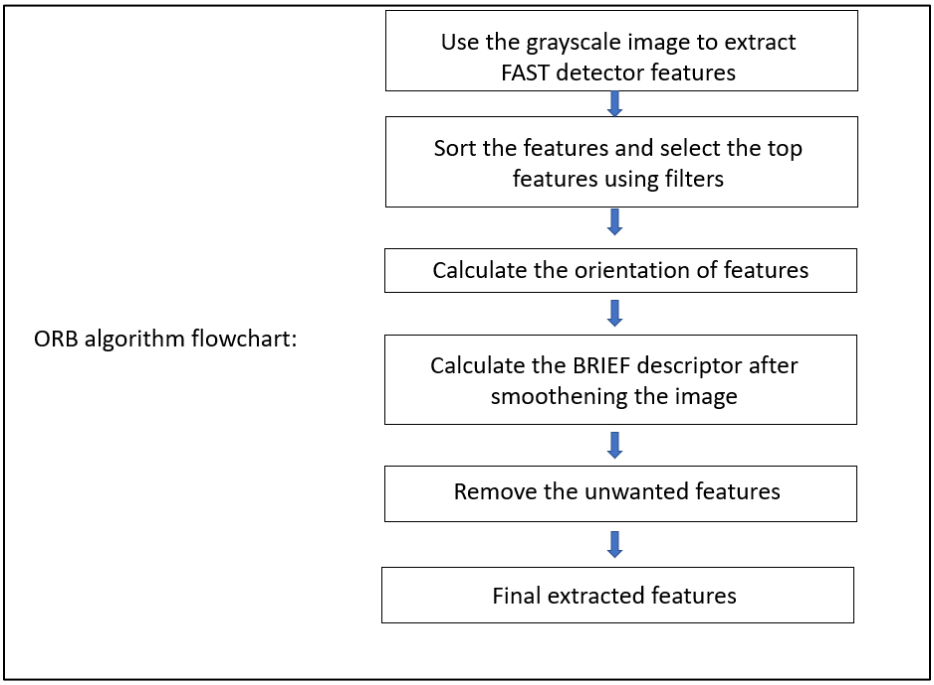
used to test 4 different LSTM models and then their accuracy was interpreted by comparing the predictions with true labels.

LSTM models overcome the problems of a normal Recurrent Neural Network. The problem with RNN is vanishing and exploding gradient caused by backward propagation step. LSTM trains the model using sequences that can utilize up to hundreds of time steps which RNN alone struggles to do. LSTM consists of gates, and it is the responsibility of gates to decide which information stored in the cell is relevant. This ensures that there is no problem of exploding or vanishing gradient. The best model in this project is Multilayer Bi-LSTM with scehaving the highest accuracy.

Oriented FAST and Rotated BRIEF(ORB) is used, and the machine learning model used is k-Nearest Neighbors. The output is the accuracy with which certain emotions can be predicted. Because ORB is free to use for commercial purposes, it was chosen above scale-invariant feature transformation (SIFT) and speeded up robust features (SURF). The number of features chosen by the ORB algorithm can be adjusted manually to make identification and detection easier as shown in figure 6. Even though SURF and SIFT can detect more characteristics than ORB, ORB is faster than all the algorithms. Using Oriented FAST and Rotated BRIEF, the k-Nearest neighbor classification algorithm is slightly modified to consider the similarity between the facial features rather than the images alone. The facial key points will be selected from the most consistent points on the face. This will help in identifying the basic structure and the outline. By using this as the baseline model, the intensity of emotions can be measured. Figure 58 shows Flowchart of ORB algorithm.

Figure 58

Flowchart of ORB algorithm



Note. Proposed flowchart to extract features using ORB and feed the output vector as input to the KNN algorithm.

4.2. Model Supports

In this project, we're using machine learning and deep learning algorithms to extract the facial features and classify the images for each emotion, which can be very intensive computationally. The volume of the training, validation and testing datasets for this project are also substantially high. The datasets are stored and extracted from Google drive. Alternatively, Amazon's S3 bucket, Google Cloud Storage, or any Cloud based storage platforms can be used to store and retrieve the data. For the DAiSEE dataset (Gupta et al., 2018) used in this project around 15GB of storage space is required to store the initial videos, and an additional 15GB of storage space is required to store the extracted and transformed image frames.

For coding of model implementation, ideally a cloud-based IDE and a large cluster with 32GB memory, 8-16 worker nodes and a general-purpose SSD is preferred. If working with a non-cluster-based IDE, a processing system with 8GB GPU, 8 core CPU, 32GB RAM and 256 GB SSD is preferred.

A programming language with predefined machine learning packages, libraries and the ability to process huge datasets is required. For this project, we are using Python as the programming language. To implement the chosen machine learning algorithms, packages including but not limited to pandas, NumPy, math, matplotlib, seaborn, PIL, CV2, imghdr, difPy, torchvision and MTCNN will be required.

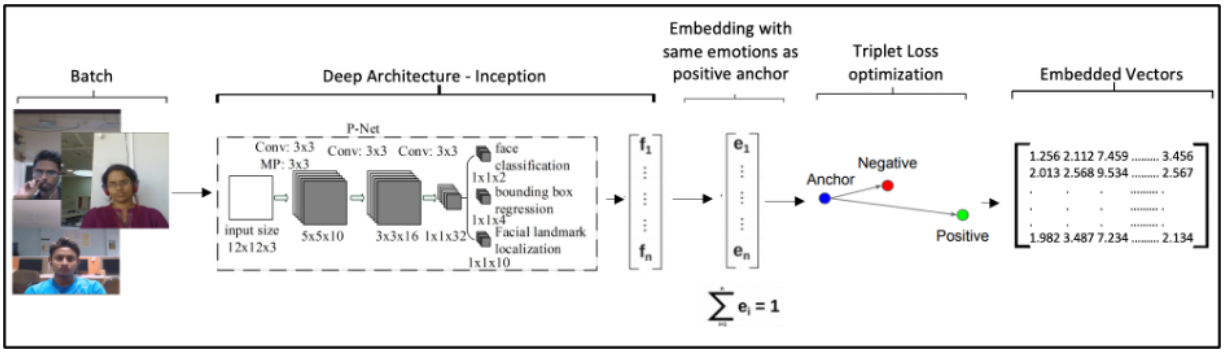
The FaceNet algorithm does not use a specific deep learning algorithm for the Deep Architecture block, and hence any relevant deep learning model can be developed to create the embeddings. In this project, we use the Inception network from GoogleNet (Szegedy et al., 2014) as the deep architecture. The network consists of a 22 layer deeply connected network, which includes 1x1 convolutional layers for dimensionality reduction and learning patterns from the depth of the images, sparse convolutional layers with 3x3 and 5x5 filters to learn spatial patterns, a max pooling layer with 32 filters, and a concatenation layer. The convolutional layers use the Rectified Linear Unit activation function. The traditional FaceNet algorithm is designed to classify images based on the similarity between images of the same person. However, our use case is to classify emotions, and not faces. So, the parameters and algorithm need to be adjusted in such a way that the embedding vector is optimized to identify the emotions. This can be done by using the following adjusted FaceNet algorithm (Figure 7) to find the optimized loss function parameters:

1. Choose an arbitrary image as the anchor image

2. Arbitrarily choose another image portraying the same emotion, to act as the positive image for the anchor image
3. Arbitrarily choose an image portraying a different emotion, to act as the negative image for the anchor image
4. Set the FaceNet loss function parameter α such that the anchor and positive image are closer together least and the anchor and negative image lie farther apart
5. Steps 1-4 are repeated by adding more and more images as anchor, positive and negative images, and optimizing α at each step

Figure 59

Adjusted FaceNet Architecture



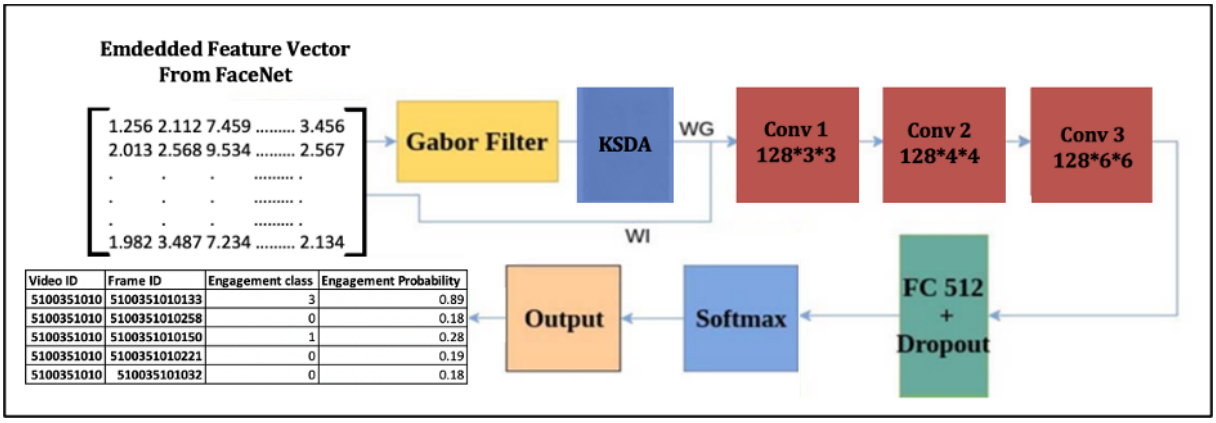
Note. Adjusted FaceNet Architecture

The traditional EmotioNet algorithm identifies the key facial landmarks and generates the facial feature vector in the Euclidean space using these landmarks. In this project, instead of using these landmarks to create the feature vector, the extracted features from FaceNet are fed to the algorithm. The subsequent steps to classify the images based on emotions using EmotioNet are like what is mentioned in the Model Proposals section. The number of layers for the Residual Neural Network are experimented with, and the best performing combination of layers and

neurons are chosen. In this project, the best performance was observed with three convolutional layers and one Fully Connected layer with dropout. First is a 3x3 convolutional layer with 128 filters, followed by 4x4 and 5x5 convolution layers with 128 filters each. All convolution layers use the ReLU activation function. The Fully Connected layer has 512 neurons and outputs the probability of each emotion in their respective models. The architecture of the adjusted EmotioNet algorithm for Engagement Emotion is represented in Figure 59. Similarly, three more versions of the same model will be developed, each trained specifically for Boredom, Confusion and Frustration, respectively.

Figure 60

Adjusted EmotioNet Architecture



Note. Adjusted EmotioNet Architecture

68 facial landmark identification models utilizing Dlib's library identifies and depicts prominent areas of the face. Facial extraction is done in two steps, firstly, face detection by identifying a person's face which returns values of rectangle vectors in x,y,w,z coordinates. This helps reduce the computational cost. Secondly, the Euclidean distance is used to determine the distances from the center point to each other data point, as well as the related coordinates to map

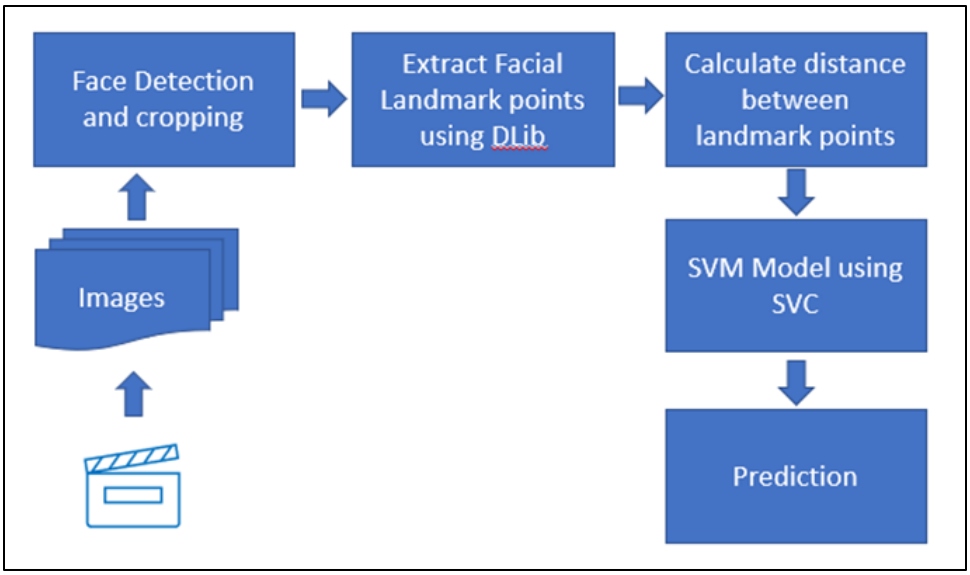
the facial features. The resultant vectors are then fit into SVM- SVC classifier for final classification. Figure 61 provides an overview of the model.

CNN using Kera's sequential model with Leaky ReLu is used in another approach to extract features. It consists of convolutional, pooling, and fully connected layers. We have used Conv2D layers with 32 and 64 layers. Leaky ReLu is selected as the activation method upon an input image resized to 48*48. Softmax is used for activation and dense output layer have been set to four corresponding top four intensities of a particular emotion. Most used Adam optimizer is used in this case with cross entropy loss function to evaluate the accuracy. The vector output is fit into SVM-SVC classifier for each emotion and then concatenated to provide overall results.

SVM does not support multi class classification. SVC with linear kernel is used from LIBSVM library. Each emotion class is modelled separately, and intensity levels (0,1,2,3) are broken down into binary classification using one -to -one approach where a hyperplane between two intensity levels is drawn ignoring the other levels. The results are then combined to provide final emotion classification. This method helps us better understand the loss function during training and validation.

Figure 61

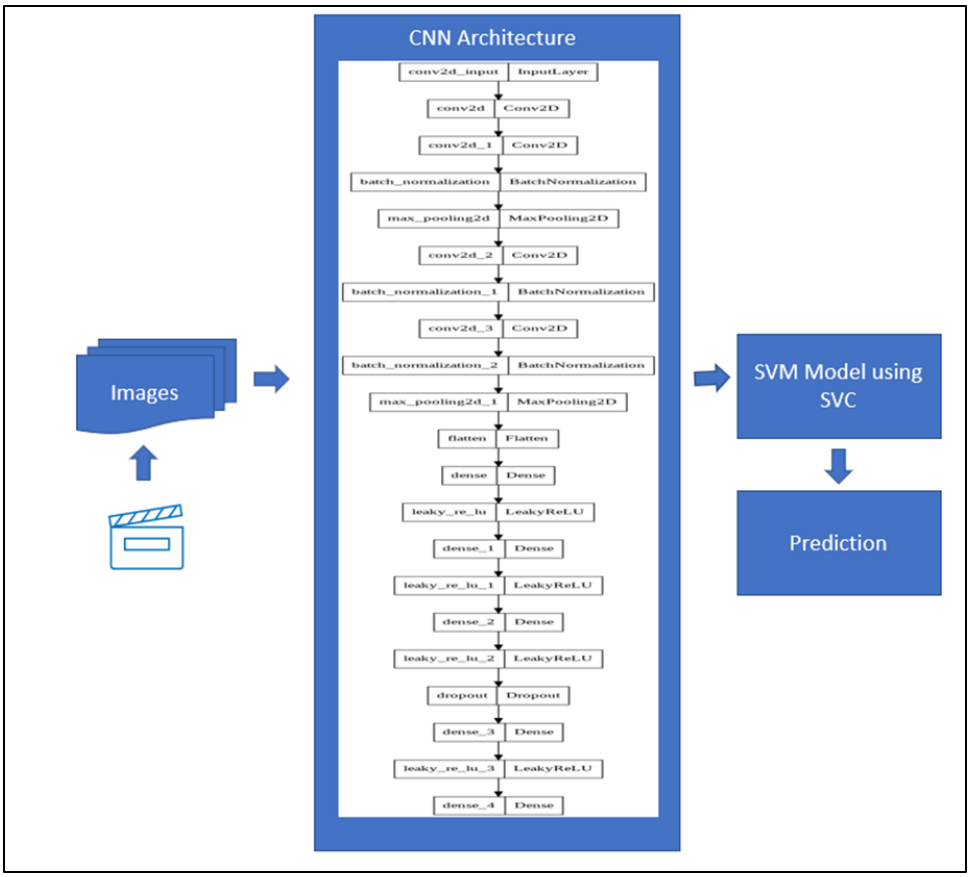
68 Facial Landmark Detection architecture with SVM Model



Note. Pictorial representation of model implementation flow and architecture

Figure 62

CNN Facial Feature Extraction architecture with SVM Model

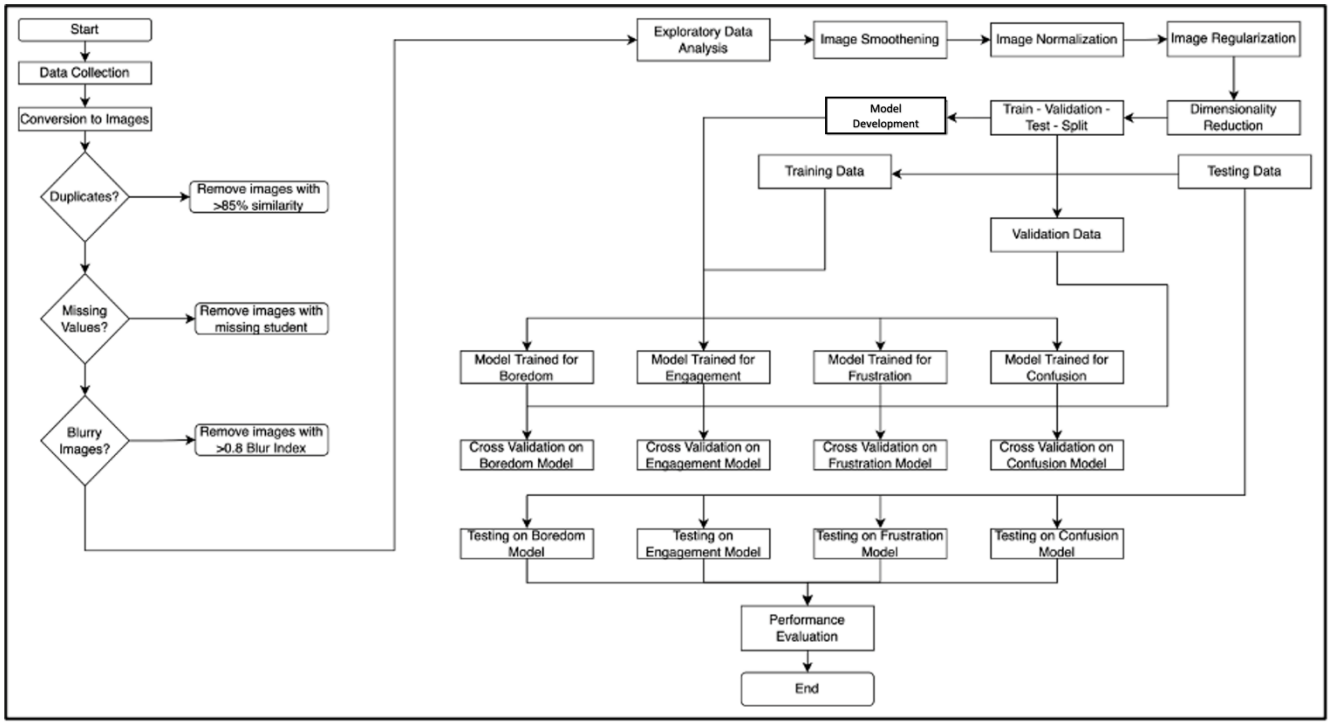


Note. Pictorial representation of model implementation flow along with CNN architecture

The flowchart in Figure 63 depicts the detailed process flow from data collection to classification of images.

Figure 63

Data and Process Flow of the Project



Note. Data and Process Flow of the Project

In this project we also implemented 4 LSTM models and various preprocessing techniques were explored that led to improvement in accuracy of prediction. Four different LSTM models namely: single LSTM, stacked LSTM, Bi-LSTM and Multilayer Bi-LSTM were used and the corresponding performance was compared.

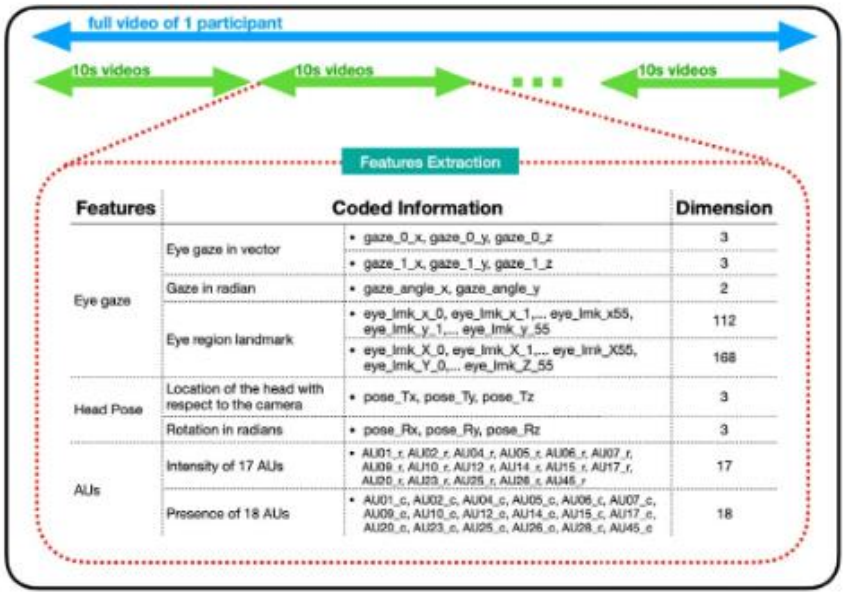
For feature extraction OpenFace is used. OpenFace is open-source toolkit used for facial landmark detection. It can also be used to estimate head pose, to recognize facial action unit and to estimate eye gaze.

Gazing of eyes, Head Pose and AUs are very important features for emotion recognition. Eye gaze is deeply linked with attention of a user, when a user fixes his gaze over screen that means he is paying attention to that content. Similarly head pose of a user is also important to

determine their engagement level. Action units are basic actions of individual muscles or group of muscles

Figure 64

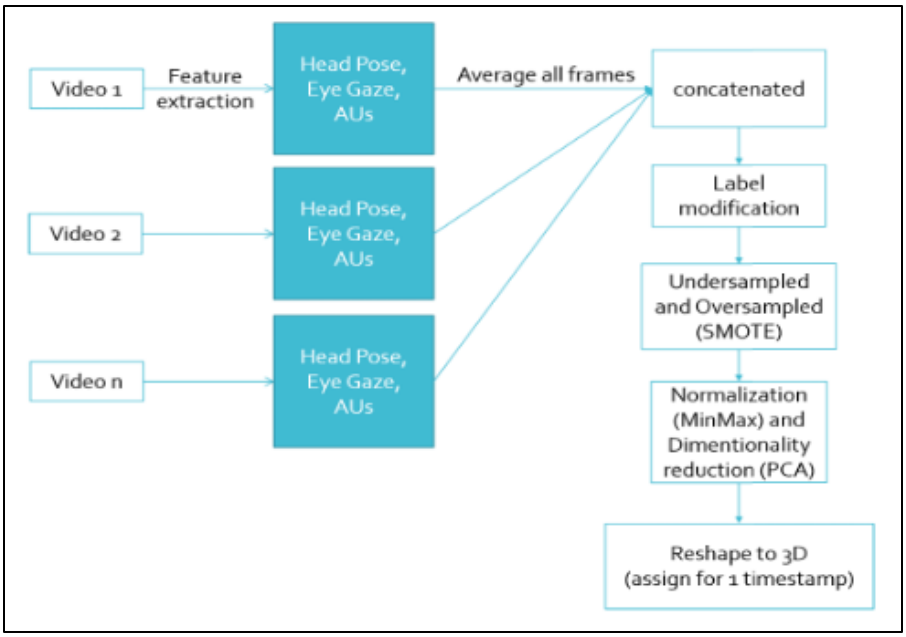
Feature extraction for one participant in LSTM



Note. Feature extraction for one participant adapted from "Nur Karimah, S., Unoki, T., & Hasegawa, S. (2021). Implementation of long short-term memory (LSTM) models for engagement estimation in online learning. 2021 IEEE International Conference on Engineering, Technology & Education (TALE) (https://doi.org/10.1109/tale52509.2021.9678909).

Figure 65

Data pre-processing in LSTM



Note. Data pre-process adapted from "Nur Karimah, S., Unoki, T., & Hasegawa, S. (2021). Implementation of long short-term memory (LSTM) models for engagement estimation in online learning. 2021 IEEE International Conference on Engineering, Technology & Education (TALE) (https://doi.org/10.1109/tale52509.2021.9678909).

Head Pose, gazing of eye and AU are fetched from the video, the idea is to average all the frames. 30 FPS was the rate of the video. The resulting frame from the averaged value of all the frames in a video are considered as single time step in sequence data. Labels were incorporated to this averaged frame, and it was then concatenated with other avg frames of all videos in a particular file. Instances in Non engaged class were very less as compared to other class and hence techniques such as under sampling and oversampling both were used for training data. Instances of the class more engaged and nominally engaged were undersampled by 50% and then instances for the class not engaged were oversampled using SMOTE.

Min-max normalization was used to rescale. This was done to decrease the dimension of the dataset while keeping as much information as possible and later PCA was used.

In order to find best preprocessing technique, experiments were conducted using below six scenarios by applying resampling in all the cases except the sixth one.

Table 4

Dataset processing techniques

Scenario 1	No normalization, no PCA.
Scenario 2	Only apply normalization.
Scenario 3	Only apply PCA with N = 250.
Scenario 4	Apply PCA first, then normalization.
Scenario 5	Apply normalization first, then PCA.
Scenario 6	Apply scenario 5 with the original data distribution (no undersampling nor oversampling).

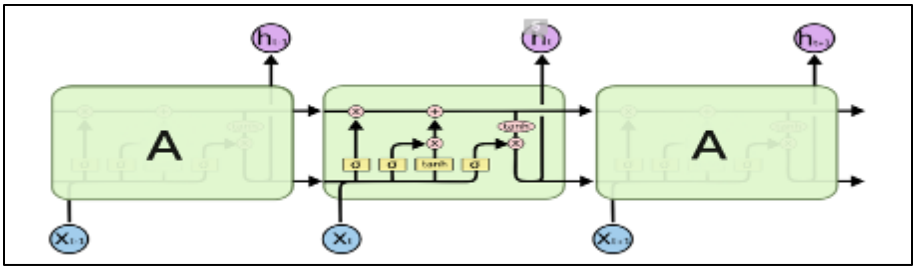
Note. Dataset processing techniques adapted from "Nur Karimah, S., Unoki, T., & Hasegawa, S. (2021). Implementation of long short-term memory (LSTM) models for engagement estimation in online learning. 2021 IEEE International Conference on Engineering, Technology & Education (TALE) (https://doi.org/10.1109/tale52509.2021.9678909).

In this project emphasis is also given on data processing techniques and experiments were conducted in total 6 scenarios. In first scenario, no normalization and no PCA was done. In second scenario normalization was used. In third scenario, PCA was used with total 250 features. In fourth scenario, PCA was first applied followed by normalization. In fifth scenario, normalization was applied followed by PCA. In sixth scenario, normalization was applied

followed by PCA with the original data that means no oversampling or undersampling was done. The aim is to find the best models using these different scenarios. All the 4 LSTM models were tested in these different types of scenarios and the corresponding results was gathered.

Figure 66

LSTM Diagram



Note. LSTM Diagram adapted from "How the LSTM improves the RNN by Tiago.M, Jan 1,2021

LSTM, a type of RNN learns from long order dependency in sequence prediction problems. Its gates help in regulating the flow of information through the unit.

4.3. Model Comparison and Justification

The Inception algorithm is basically a 22-layer Convolutional Neural Network that extracts the faces out of the entire image and learns the facial features. FaceNet uses the Inception output, and performs L2 regularization on the extracted components, and then embeds them into Euclidean space. While embedding the facial features into Euclidean space is a fairly simple process, the Triplet loss optimization to ensure that the images with the same emotions are closer together than the others is a computationally intensive, as each image is compared against every other image. The computational intensity of this can be compared to that of a kernel matrix computation.

Feeding the feature vectors from FaceNet as the Euclidean vectors for EmotioNet is experimented in this paper. The effective embedding and feature extraction considering Emotion

detection as the key in choosing the anchor images of FaceNet, along with the capability of EmotioNet to effectively classify non-linear data clearly stands out in the performance evaluation. The advantages of the FaceNet with EmotioNet algorithm are its ability to generalize well on non-linear data and identify even minute differences in expression changes. The DAiSEE dataset (Gupta et al., 2018) is proven to be challenging with very minute differences in expression between multiple emotions. One of the limitations of the proposed algorithm is the increased computational complexity, as kernel operations need to be performed on the Euclidean embedded vector.

In this project, we are applying two different facial feature extraction techniques separately in machine learning SVM model to evaluate the performance of each model on DAiSEE dataset. SVM classification model trained using 68 facial landmark detection using Dlib library with calculating distance between data point using Euclidean method is experimented. Where handcraft features identify the face in an image removing the irrelevant features present in the input helps reduce the computational cost of the feature extraction model. Furthermore, calculating distances using Euclidean measure provided recognition of emotions and their intensities with increasing cost. In the second technique, when SVM classification model is trained using custom CNN facial feature recognition method with activation function Leaky ReLU with linear kernel. The linear combination of the convolutional filter-weighted pixels pulls certain form of information from the picture. Linear kernel helps extract important characteristics in a picture as they are generally localized, therefore applying convolutions to neighborhood pixels at a time is acceptable. For accuracy, the kernel size is maintained modestly. Use of Leaky ReLu activation helps improve the training time and cost of the model with a compromise in non-linearity in exchange for greater gradient back propagation. Therefore, overall performance and computational complexity is better handled in custom CNN compared to the handcraft method.

Unlike common deep learning systems where computational techniques are black box, SVMs allow for some intuitive thinking and insight. They cope with unbalanced data and overfitting by permitting certain prediction errors on the training images. Even in high dimensional datasets, the model handles data that is linearly inseparable using kernels such as polynomial kernels and RBF kernel. A hierarchy of binary classifiers integrated together can perform multi-class and multi label classification. In a variety of uses, they presently outperform deep neural networks with the right hyperparameter tuning.

Multilayer Bi-LSTMs recurrent nature helps these models to remember pieces of information that it has seen earlier in the sequence whether it is forward direction or backward direction. Bi-LSTM captures the flow of information in both directions. Multilayer Bi-LSTM differs from single layer LSTM because in Multilayer Bi-LSTM the output is also used as an input for the next layer. In the case of Multilayer Bi-LSTM the output is concatenation of forward and backward direction, this is the reason Bi-LSTM are complex in nature. In Bi-LSTM sequence of information flows in both directions and that is how it's different from normal LSTM. In our case timesteps of the input sequence are available and Multilayer Bi-LSTM will train multiple layers of LSTM in both forward and backward direction that results in improvement of the model's performance.

LSTM minimizes the effect of vanishing gradient, but it doesn't eliminate it completely. With the continuous movement of data across the cell, it becomes more and more complex. LSTM requires a lot of time and resources to get trained and deploy in real-world applications. Due to linear layers, it requires high memory bandwidth, so if we consider hardware requirement then LSTM are not that great. Though LSTM remembers information, research is still going on to find something more concrete through which these models remember information for a longer

duration of time. LSTMs are greatly affected by different random weight allocation. LSTMs tend to overfit and here it is difficult to incorporate dropout algorithm.

ORB works on the key point matching idea, where a binary vector is constructed for each key point to reduce computational time. This will aid in the reduction of time complexity. It is a hybrid of the FAST keypoint detector and the BRIEF descriptor, with numerous enhancements. These modifications are known as oFAST and rBRIEF, respectively. It is also inexpensive and comes with an impressive computational performance. The rBRIEF used is very resistant to noise, blur, and low lighting. rBRIEF is very sensitive to rotation and underperformed. FAST algorithm lacks the features of an orientation operator and produces only a single feature. This is why oFAST(oriented FAST) was introduced. Similarly, BRIEF is rotation variant. Whenever a given image is rotated, it lacks the ability to map and find the key points.

The output of ORB is feature vectors which are fed as the input for k-Nearest Neighbor. The advantage of KNN is there is no training step, and it constantly evolves to new data since it’s instance-based learning. While the advantages are prominent, it has its own downside since the curse of dimensionality creeps in as the dataset increases. It also distorts the optimal k-value which makes classification go offside.

When comparing the machine learning algorithms used in this project, the instance-based learning capability of KNN irrespective of data linearity makes it better than SVM. Also, computationally, calculating the dxd kernel matrix in SVM will be more intensive than calculating distances in KNN. When comparing the deep learning models, Bi-LSTM's ability to train multiple layers of LSTM in both forward and backward direction when performing classification on time-sequence data serves as an added advantage when compared to EmotioNet.

4.4. Model Evaluation Methods

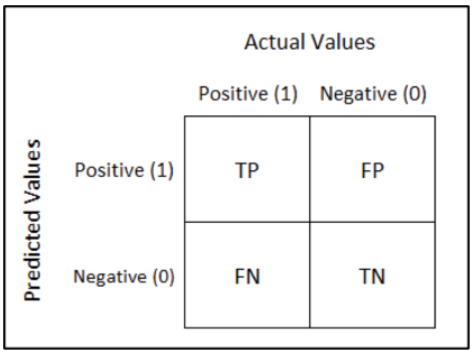
We have developed four binary classifiers, each predicting the intensity of the respective emotion portrayed by students in the images. To evaluate the prediction performance of binary classifiers, several metrics such as Accuracy, Precision, Recall, F1 Score, etc. are available. All these metrics are derived from the confusion matrix, which represents the count of correctly classified images, against the misclassified ones. The following subsections explain the model evaluation metrics used in this paper.

4.4.1. Confusion Matrix

The Confusion matrix is a matrix form representation of the correctly and incorrectly classified images. In this project, True Positives (TP) are those images that according to the annotation or label portray a particular emotion and are predicted to have portrayed the same emotion by the model. True Negatives (TN) are those images which actually portray an emotion, but the model predicts that it does not portray that emotion. False Negatives (FN) are those images where according to the label a particular emotion is not portrayed, and the model also predicts that the same emotion is not portrayed in the image. False positives (FP) are those where a particular emotion is not portrayed according to the label, but the model predicts that the emotion is portrayed (Kulkarni, 2020). Figure 12 represents a typical confusion matrix.

Figure 67

A Typical Confusion Matrix



Note. A Typical Confusion Matrix from "Understanding Confusion Matrix" by Narkhede Sarang (2018), *Towards Data Science* (https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62)

4.4.2. Accuracy

Accuracy represents the percentage of images for which the emotions were correctly identified (Vothihong, 2017). It is given by the formula (TP+FN) / (TP+TN+FP+FN).

4.4.3. Precision

Precision is the percentage of images that were predicted to portray an emotion and portray the same emotion, amongst all images that were predicted by the model to have portrayed that emotion. It is given by the formula TP/(TP+FP) (Vothihong, 2017).

4.4.4. Recall

Recall, also called the true Positive Rate, gives the percentage of positives that were correctly identified by the model. Here, it is the percentage of images that were predicted to portray an emotion and portray the same emotion, amongst all images that portrayed that emotion. It is given by the formula TP/(TP+FN) (Vothihong, 2017).

4.4.5. F1 Score

F1 score represents the harmonic average between Precision and Recall (Ng, 2018). In datasets with imbalanced classes, F1 score is an important statistic that represents the model performance. It is given by the formula 2*(Precision * Recall) / (Precision + Recall).

All the above-mentioned metrics are used for comparing the performance of the four developed models amongst one another. For comparison of the model efficiency with existing benchmarked models, accuracy is primarily used. Since certain existing models predict only whether an emotion is portrayed or not, and not the intensity at which it is portrayed, two classes based on the probabilities, with 0.5 as the threshold is used to map the probability of the emotion to binary classes. The performance metrics are hence calculated based on the binary classes for each emotion.

4.5. Model Validation and Evaluation

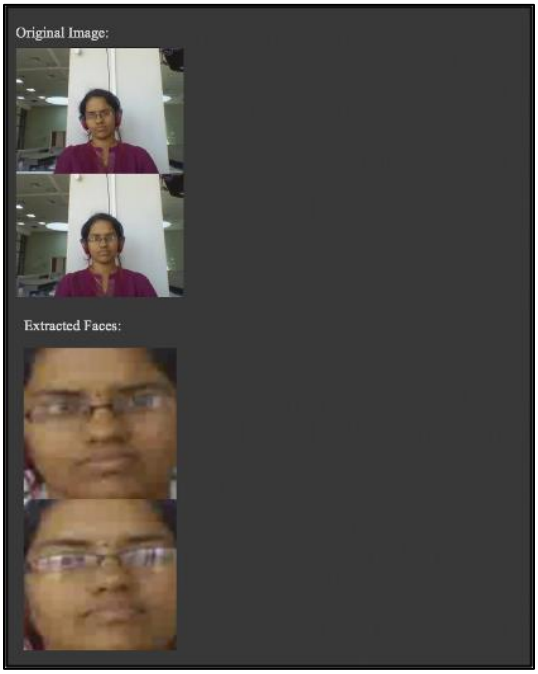
DAiSEE dataset consists of separate folders for train, validate and test dataset. As part of the project, the test folder was not utilized as it was not annotated. After converting videos into image frames and preprocessing, 80% of training images were used for training the model and 20% for testing. Validation folder images were used for validation purposes.

The proposed FaceNet with EmotioNet algorithm from section 4.2. is trained separately for each emotion and correspondingly, four model runs are done. The developed models are cross validated against the validation dataset and tested using the test dataset provided separately by DAiSEE (Gupta et al., 2018).

Figure 16 shows a sample input image and the corresponding intermediate result from FaceNet after the face extraction and Figure 17 shows a sample Euclidean vector after FaceNet's embedding process.

Figure 68

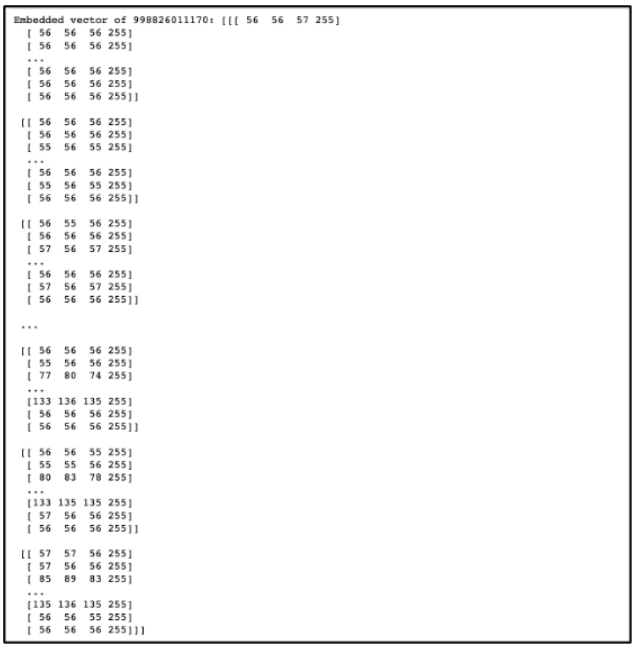
Original and extracted face from testing dataset samples



Note. Original and extracted face from testing dataset samples

Figure 69

Euclidean Vector output from FaceNet on testing dataset samples



Note. Euclidean Vector output from FaceNet on testing dataset samples

The predictions from the four models developed for the four emotions are evaluated against the actual emotions as per the label information provided in the DAiSEE dataset using the performance metrics specified in the above section. To that end, Table 2 depicts samples from results of one of the developed models when tested against test dataset of DAiSEE.

Table 5

Sample Results from Engagement Classifier using EmotioNet Algorithm

Video ID	Frame ID	Engagement class	Engagement Probability
5100351010	**5100351010133**	3	0.89
5100351010	**5100351010258**	0	0.18
5100351010	**5100351010150**	1	0.28

5100351010	**5100351010221**	0	0.19
5100351010	**510035101032**	0	0.18

Note. Sample Results from Engagement Classifier using EmotioNet Algorithm

Table 6 and Figure 70 represent various performance metrics derived from the results for each emotion for the EmotioNet model. We can observe that the proposed algorithm works best in classifying Frustration against the other emotions, while the least performance is seen for predicting Boredom.

Table 6

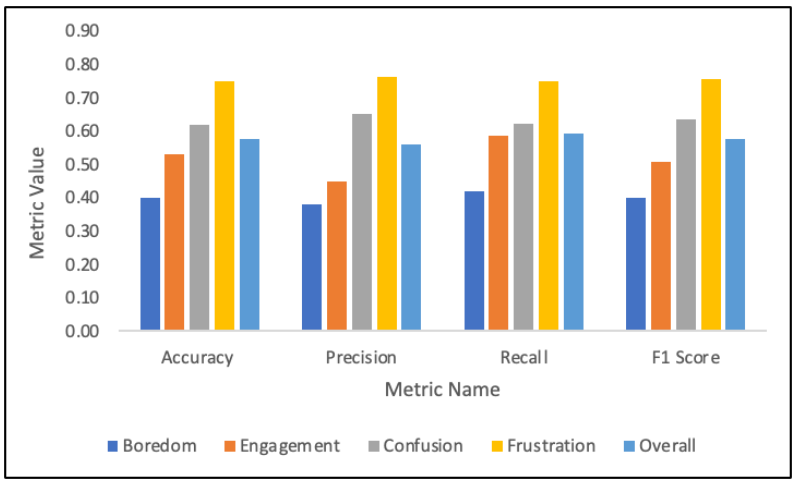
Performance Metrics of Developed Models using EmotioNet Algorithm

Emotion	Accuracy	Precision	Recall	F1 Score
Boredom	0.40	0.38	0.42	0.40
Engagement	0.53	0.45	0.59	0.51
Confusion	0.62	0.65	0.62	0.64
Frustration	0.75	0.76	0.75	0.76
Overall	0.58	0.56	0.59	0.58

Note. Performance Metrics of Developed Models using EmotioNet Algorithm

Figure 70

Performance Comparison of Developed Models using EmotioNet Algorithm



Note. Performance Comparison of Developed Models using EmotioNet Algorithm

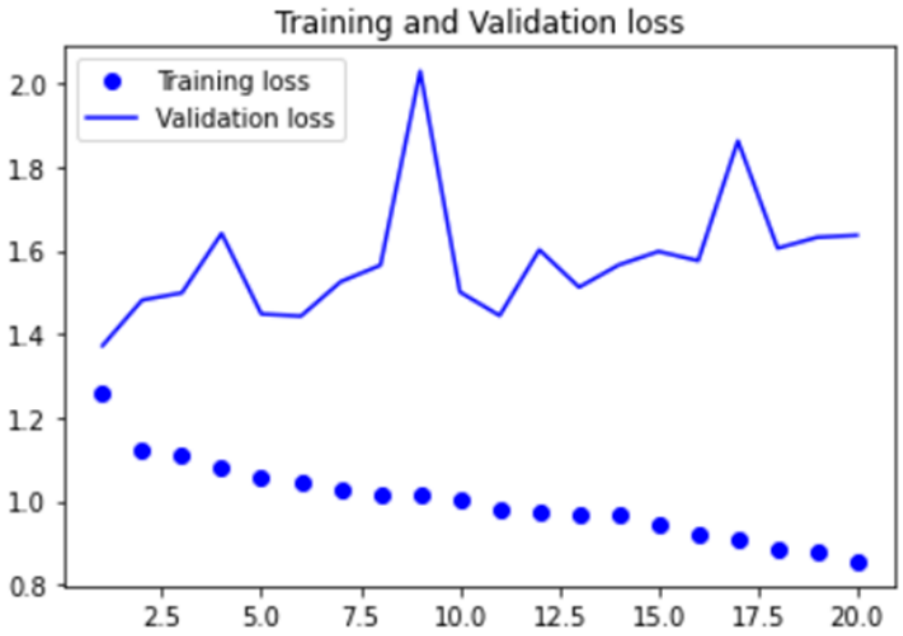
Overall, based on all the depicted performance metrics, the classifiers are best capable of identifying Frustration and Confusion, and it understandable because more facial muscle movement involved to portray these emotions and hence more variation exists from a neutral expression. Boredom and Engagement expressions have less variation from the neutral face and hence are more difficult to capture. When comparing the FaceNet with EmotioNet model performance on the DAiSEE dataset against the benchmarked results of EmotioNet on the DAiSEE dataset as in Figure 10, the FaceNet with EmotioNet is better able to identify all the emotions.

For custom CNN based technique of SVM modelling, the training and validation loss was evaluated. The training loss provides the measure of performance of training data fit in the model while validation loss provides the model's measure to fit testing data. We have used L2 regularize to generalize the dataset. Adam optimizer, gradient descent methodology is used to reduce noise and adjust weights and pace of learning to minimize the loss. Figure 71, shows the graph epoch 20 and batch size 64 with validation done for each iteration. Since modelling has

been done on sample data due to computational complexities, we can see that validation loss is greater than training loss which shows probability of under fitting scenario.

Figure 71

Training and Validation Loss Diagram

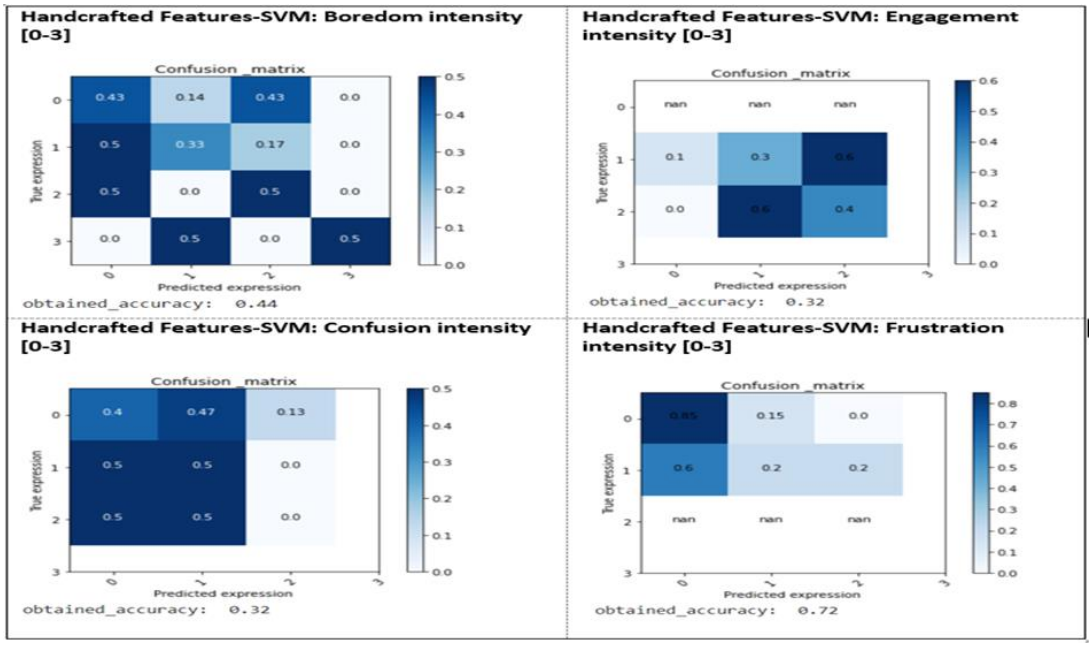


Note. Graphical representation of loss curved while implementing CNN model over multiple iterations

Figure 72 shows confusion matrix of each model created for each emotion using handcraft feature extraction technique. As shown in the figure, accuracy of frustration is higher than other emotions at 72% with higher accuracy for extremely low and low intensity followed by boredom at an accuracy level of 44% with even accuracies at all intensity levels. Accuracy of engagement and confusion is low at 32%. Therefore, overall accuracy achieved by combining four models to predict final engagement classification using 68 facial landmarks detection with Dlib averages to 45%.

Figure 72

Confusion Matrix of FRE using 68 Facial Landmarks with Dlib and SVM

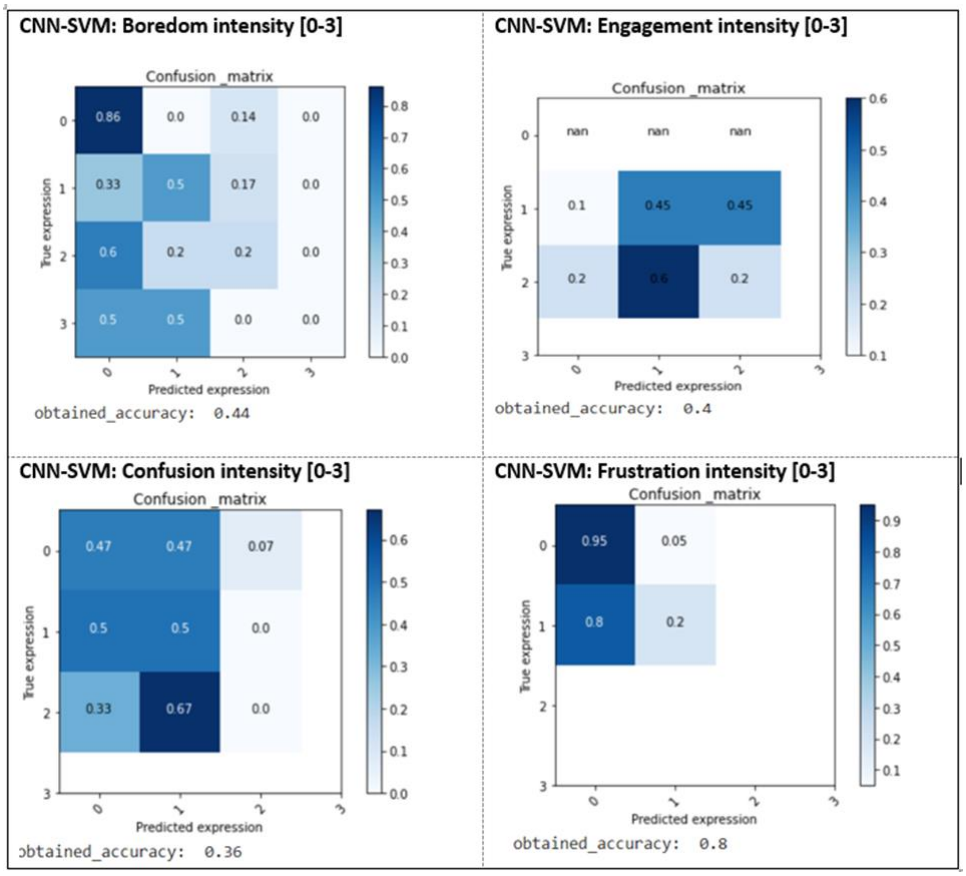


Note. The graphs shown are for each emotion for which modelling is done separately to classify accuracy across four different intensity levels

Figure 73 shows confusion matrix of each model created for each emotion using custom CNN facial feature extraction technique and then SVM algorithm used for classification. As shown in the figure, accuracy for frustration is high at 80% higher accuracy for very low and low emotion intensities followed. Boredom has an accuracy of 44% evenly distributed across all intensity levels closely followed by engagement at 40% and lowest for frustration at 36%. Therefore, overall accuracy achieved by combining four models to predict final engagement classification using custom CNN technique averages 50%. Table 7 shows Overall evaluation metrics of SVM models that were implemented.

Figure 73

Confusion Matrix of FRE using custom CNN and SVM



Note. The graphs shown are for each emotion for which modelling is done separately to classify accuracy across four different intensity levels

Table 7

Overall evaluation metrics of SVM models implemented

Model\Evaluation	**Accuracy**	**Precision**	**Recall**
68 Facial Landmarks Detection with SVM	0.45	0.45	0.45
CNN with SVM	0.5	0.5	0.5

Note. The table demonstrates a comparison of results of accuracy, precision and recall for models

Overall accuracy of custom CNN model is slightly better than handcraft models which is expected. We can also see that CNN technique is better in classifying engagement emotion compared to other techniques while confusion classification is least accurate irrespective of technique used. Precision and recall show related results as accuracy as mentioned in table 7.

Figure 74 shows the generated image frames from original data. Experiments were also conducted using 4 different LSTM models. Results of experiments were summarized using mean squared error loss function and Adam with 1e – 3 learning rate and trained in 150 epochs.

Figure 74

Generated Image Frames from original data

Note. Generated Image Frames from original data

From Table 8 we can conclude that Multilayer Bi-LSTM was the best model in terms of performance, and it has an accuracy of 0.902. Stacked LSTM is a robust model, and it shows best performance in 2,3,5 and 6 case where the accuracy is 0.627,0.627,0.800 and 0.507, respectively. Scenario 5 works decently for all the models.

Table 8

Performance Metrics of LSTM

Models	Boredom	Engagement	Confusion	Frustration	Overall
Single-LSTM	0.59	0.61	0.58	0.66	0.61
Stacked-LSTM	0.66	0.64	0.60	0.62	0.63
Bi-LSTM	0.45	0.48	0.57	0.50	0.54
Multilayer Bi-LSTM	0.88	0.85	0.91	0.96	0.90

Note. Performance Metrics of LSTM

We compared 4 different LSTM models using DAiSEE dataset to determine the engagement level of students. Greater emphasis on preprocessing steps was given. Multilayer Bi-LSTM has the best model with accuracy of 0.90, in this case the resampled data was preprocessed using PCA before applying normalization.

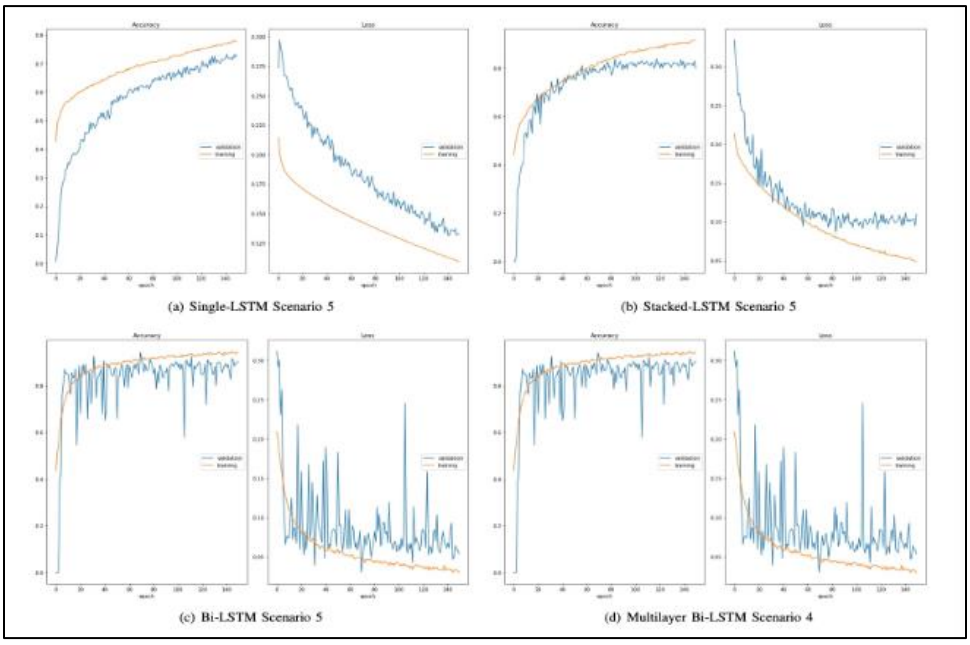
Models that have high accuracy can be implemented into online learning management systems that have cameras. This can be very useful to educators and online LMS, they can utilize this to find out when the student is more focused. With engagement recognition educators can identify in which topics students are interested and when their attention starts drifting, they can probably identify their core areas of strengths and weakness and work accordingly to improvise the content where students are not engaged fully. This can be further extended in online classes where the tutor can identify the students who are not completely engaged and pay attention to such students and help these particular students to provide feedback or just explain that concept

again, by doing this student will be more focused in the classes and their dropout rate will drastically decrease. The reason for such good accuracy is because of one time step by averaging all the frames from a video. Annotators labelled the data based on the overall impression from this averaged data.

In figure 75 shows the accuracy and MSE of best performance of each model of LSTM. Out of all the LSTM models Multilayer Bi-LSTM was the best model with accuracy of 0.902.

Figure 75

Accuracy and MSE of best performance of each model



Note. Accuracy and MSE of best performance of each model

In Figure 76, The green dots are the recognized facial features that need more hyper-tuning and modifications to accurately extract the features

Figure 76

Output of the image when ORB is used to extract facial features.

Note. The green dots are the recognized facial features that need more hyper-tuning and modifications to accurately extract the features.

The posture of the person also determines the emotion. If the person is sitting in a laid-back position, it often correlates to more boredom. And if a person is sitting in an upright position, it often correlates to him/her being more attentive. Once the features are extracted as vectors as shown in figure 25, the KNN algorithm is chosen because it is fast and easy to implement. All the features will be extracted over the points of interest like nose, eye position, eyebrows, and mouth. It counts the k-nearest neighbors by majority vote. Here the distance metric can range from Manhattan, Minkowski to Euclidean. These distance metrics can be adjusted and tweaked to improve the performance of the model. The similarity measure score is calculated depending on the number of nearest neighbors.

Table 9 depicts the summarized results of overall accuracy from all the models developed. We can observe that all the algorithms perform best in predicting Frustration and the worst in predicting Boredom and Confusion. Multilayer Bi-LSTM performs the best of all the other models, with an overall accuracy of 90%. When comparing deep learning models versus machine learning models, for the DAiSEE dataset, both the deep learning models outperform machine learning models in terms of accuracy. However, in terms of computational time and resource utilization, KNN with ORB as the feature extraction method is far more efficient than the deep learning algorithms.

Table 9

Comparison of Accuracy between all the developed models

Feature Extraction Method	Modelling Algorithm	Boredom	Engagement	Confusion	Frustration	Overall
FaceNet	**EmotioNet**	0.40	0.53	0.62	0.75	0.58
68 Facial landmark	**SVM**	0.44	0.32	0.32	0.72	0.45
CNN	**SVM**	0.44	0.40	0.36	0.80	0.50
Sonal	**LSTM**	0.88	0.85	0.91	0.96	0.90
ORB	**KNN**	0.42	0.58	0.41	0.69	0.52

Note. Comparison of Accuracy between all the developed models

References

Benitez-Quiroz, C. F., Srinivasan, R., & Martinez, A. M. (2016, December 12). *EmotioNet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild*. IEEE Xplore. Retrieved May 15, 2022, from https://ieeexplore.ieee.org/abstract/document/7780969

Du, S., & Martinez, A. M. (2014, March 31). *Compound facial expressions of emotion: From basic research to clinical ...* PNAS. Retrieved May 15, 2022, from https://www.researchgate.net/publication/298640956_Compound_facial_expressions_of_emotion_From_basic_research_to_clinical_applications

Gupta, A., D'Cunha, A., Awasthi, K., & Balasubramanian, V. (2018, April 13). *Daisee: Towards user engagement recognition in the wild*. arXiv.org. Retrieved May 15, 2022, from https://arxiv.org/abs/1609.01885

Herawan, T., Mat Deris, M., Nawi, N. M., & Ghazali, R. (2016). *Recent advances on Soft Computing and data mining. SpringerLink. Retrieved May 15, 2022, from https://link.springer.com/book/10.1007/978-3-319-51281-5*

Journal, I. R. J. E. T. (2018, July 28). *IRJET-V5I6191.pdf*. Academia.edu. Retrieved May 14, 2022, from https://www.academia.edu/37138364/IRJET_V5I6191_pdf

Krzanowski, W. J. (2009). ROC Curves for Continuous Data. O'Reilly Online Learning. https://www.oreilly.com/library/view/roc-curves-for/9781439800225/

Kulkarni, A., Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy. In data democracy (pp. 83-106). Academic Press.

Lucey, P., Cohn, J. F., & Kanade, T. (2010, August 9). *The extended cohn-kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression*. IEEE Xplore. Retrieved May 15, 2022, from https://ieeexplore.ieee.org/document/5543262

Lu, C., & Tang, X. (2014, December 20). *Surpassing human-level face verification performance on LFW with Gaussianface*. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/1404.3840

Mavadati, S. M., Mahoor, M. H., Bartlett, K., & Trinh, P. (2013, March 7). *DISFA: A spontaneous facial action intensity database*. IEEE Xplore. Retrieved May 15, 2022, from https://ieeexplore.ieee.org/document/6475933

Michel, P., & Kaliouby, R. E. (2003, November 1). *Real time facial expression recognition in video using support vector machines: Proceedings of the 5th International Conference on Multimodal Interfaces*. ACM Conferences. Retrieved May 14, 2022, from https://dl.acm.org/doi/abs/10.1145/958432.958479?casa_token=fCs6gH_KryMAAAAA%3Ai9ENUAY8DUKRexd4AwIdUhj0aeb7RLZfvll3UDxpOpjXT70qVCzD-PleYPA9siteGvJrEM_QyyjDHw

Neth, D., & Martinez, A. M. (2009, January 1). *Emotion perception in emotionless face images suggests a norm-based representation*. Journal of Vision. Retrieved May 15, 2022, from https://jov.arvojournals.org/article.aspx?articleid=2122685

Ng, A. (2018). The Complete Machine Learning Course with Python. O'Reilly Online Learning. https://www.oreilly.com/videos/the-complete-machine/9781789953725/

Nur Karimah, S., Unoki, T., & Hasegawa, S. (2021). Implementation of long short-term memory (LSTM) models for engagement estimation in online learning. 2021 IEEE International

Conference on Engineering, Technology & Education (TALE). https://doi.org/10.1109/tale52509.2021.9678909

Schroff, F., Kalenichenko, D., & Philbin, J. (2015, June 17). *FaceNet: A unified embedding for face recognition and clustering*. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/1503.03832

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014, September 17). *Going deeper with convolutions*. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/1409.4842

Vothihong, P. (2017). Python: End-to-end Data Analysis. O'Reilly Online Learning. https://www.oreilly.com/library/view/python-end-to-end-data/9781788394697/

Appendix

GitHub is used for handling source code developed as part of this project implementation and to maintain the code. Python Jupyter notebook is uploaded. GitHub Link- https://github.com/manishaPaliwal/Engagement-Recognition-Using-Video-Based-Expression-Tracking

Dataset is uploaded on google drive at https://drive.google.com/file/d/1OGlbhdePFlvYjg8X4_spmaudPF0UpWZ1/view?usp=sharing