

Analyzing students' concentration in online courses through
Webcam

by

Md. Asif
19201096

Md. Imtiaz Hossain
19201031

Fouzia Sharkar
19201094

Md. Mohaimenul Islam
19201095

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2024

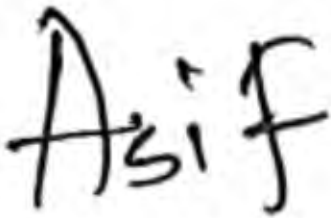
© 2024. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



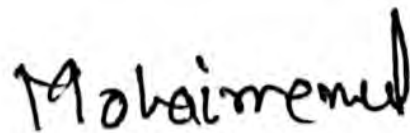
Md. Asif
19201096



Md. Imtiaz Hossain
19201031



Fouzia Sharkar
19201094



Md. Mohaimenul Islam
19201095

Approval

The thesis/project titled “Analyzing students’ concentration in online courses through Webcam” submitted by

1. Md. Asif(19201096)
2. Md. Imtiaz Hossain(19201031)
3. Fouzia Sharkar(19201094)
4. Md. Mohaimenul Islam(19201095)

Examining Committee:

Supervisor:
(Member)



Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



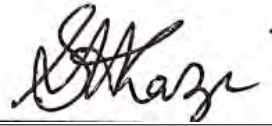
Md Tanzim Reza
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)



Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)



Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Online learning is growing in popularity these days. As a result, students typically contribute millions of course-related responses to discussion forums and exchange some learning experiences. This study focuses on online courses offered through MOOC platforms and identifies the variables that affect students' ability to stay focused. We suggest a unique method to address this issue by evaluating students' levels of concentration using the CNN architecture, MobileNetV2, VGG16, ResNet50, and InceptionV3 models. Our goal is to determine whether the issue is with students' concentration, the course material, or both. Measurement of concentration levels, evaluation of video data, comparison of model performances, and provision of class-based concentration levels (attentive, inattentive, and sleepy) are the goals of our research. The dataset underwent pre-processing, which included resizing for analysis, frame extraction, and annotation for classification.

Our research offers educators insightful information that will help them to increase the overall efficacy of online learning. Furthermore, the study advances the area by offering a methodical technique for assessing and evaluating students' concentration on online courses.

Keywords: Concentration levels; CNN; MobileNetV2; VGG16; ResNet50; and InceptionV3

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Table of Contents	v
1 Introduction	2
1.1 Motivation	2
1.2 Research Problem	2
1.3 Research Objectives	3
1.3.1 Thesis Organization	3
2 Literature Review	4
2.1 Existing Research Gaps	7
3 Methodology	8
3.1 Proposed Methodology	8
3.2 Dataset	9
3.2.1 Data requirements:	9
3.3 Data collection methodology	9
3.3.1 Recording videos	9
3.3.2 Video annotation	9
3.3.3 Splitting and classification of the videos	10
3.4 Data pre-processing	11
3.4.1 Converting to frames	11
3.4.2 Frame Annotation	12
3.4.3 Resizing images	12
3.4.4 Data Augmentation	13
3.4.5 Rescaling	14
3.4.6 Pre-processed data	15
3.5 Model Specification	15
3.5.1 Convolutional Neural Network(CNN)	15
3.5.2 MobileNet V2	17
3.5.3 VGG 16	18
3.5.4 ResNet50	18
3.5.5 InceptionV3	19

4	Implementation	20
4.1	Workflow	20
4.2	Setup for Experiment	21
4.2.1	Training Hardware and Software	21
4.2.2	Library List	22
4.2.3	Structural view of the code skeleton	23
4.3	Pre-processing & Augmentation	23
4.3.1	Pre-processing	23
4.3.2	Augmentation	24
4.4	Model selection	25
4.5	Hyperparameter Tuning	25
4.6	Design & Compile the Models	26
4.6.1	ConcentrateNet1	26
4.6.2	ConcentrateNet2	26
4.6.3	ConcentrateNet3	27
4.6.4	VGG16	27
4.6.5	ResNet50	28
4.6.6	InceptionV3	28
4.6.7	MobileNestV2	28
4.7	Train & Evaluate the Models	29
4.7.1	ConcentrateNet1	29
4.7.2	ConcentrateNet2	30
4.7.3	ConcentrateNet3	31
4.7.4	VGG16	32
4.7.5	MobileNetV2	33
4.7.6	ResNet50	34
4.7.7	InceptionV3	35
5	Result Analysis	36
5.1	Comparison between our design model (concentrateNet1,2,3)and the pre-trained model	36
5.2	Class-wise study for the Confusion matrix	37
5.3	Output	42
5.3.1	Analysing our Design model (Concentrate1, Concemtrate2, Concentrate3) and the pre-trained model	42
5.3.2	Our Design models:	43
5.3.3	Visual Representation and Analysis of the Result Implementation	46
6	Conclusion	53
6.1	Future work	53
	References	55

Chapter 1

Introduction

1.1 Motivation

We gather knowledge from both online and conventional sources concurrently. We have become very dependent on these online platforms for a better understanding of our studies. With this enthusiasm, students join many online courses but do not continue to the end which impacts their knowledge. Also, in some cases, students finish the course and achieve the certificate as well but they can not apply it in the real world which creates an anomaly. As a result, they can not achieve the goal they wanted from the course. So it is high time to find the cause behind their failure or poor outcomes.

Many students are enrolling in free online courses, and a common challenge emerges that a substantial portion tends to drop out and get poor marks. This phenomenon can be attributed to either subpar course materials or low student concentration levels. Most courses are structured around lectures, videos, PDF articles, and quizzes. As students engaged with course content, we recorded their behavior and attentiveness by capturing video footage. Using these observations, we assess how students perceive and engage with the courses. Through analyzing these features, we gain valuable insights into areas that require attention for improvement or updates, be it in course materials, instructor selection, technical aspects, or content quality. The duration of the course plays a crucial role in attention span, with research indicating that the maximum attention span for adults is around 20 minutes. This necessitates a reevaluation of both course duration and teaching methods, emphasizing interactive and captivating approaches. In our efforts to understand exam failures, we employ deep learning techniques to delve deeper into the issue. Our approach involves measuring students' concentration levels to uncover the genuine reasons behind their poor outcomes.

1.2 Research Problem

Students still struggle to feel at ease with MOOC platforms, but with time, we increasingly rely on virtual materials over traditional ones. However, a significant challenge emerges as many students enroll in courses.

MOOC platforms are actively investigating this issue to understand why students

can not do good in the exams or fail. Two common reasons come out: either the course materials or instructors fail to captivate their attention, or students struggle to concentrate on the materials. This lack of engagement leads to poor marks or even failure. In our research paper, we take a holistic approach, utilizing deep learning to measure students' concentration levels. This approach helps us pinpoint whether the issue lies with the course materials, the student's concentration, or a combination of both. By identifying areas where students tend to lose focus, educators can enhance and update their materials to provide the learners with a better learning experience.

1.3 Research Objectives

In this study, we introduce a deep learning approach to pinpoint the reasons behind students' low performance. Our main goal is to evaluate students' concentration levels, analyze the data, and determine whether students themselves are responsible for their failure, or if there's room for improvement in the course materials. This process aims to enhance the interaction between teachers and students. To teach our system to recognize student images, we utilized cutting-edge methods like Design ConcentrateNet1, Design ConcentrateNet2, Design ConcentrateNet3, VGG16, ResNet50, MobileNetV2, and InceptionV3. Additionally, this research provides an opportunity for researchers to gather extensive data and unravel the complexities of students leaving MOOC platforms. The objective of this research is:

- To measure students' concentration levels to identify the actual problem and use of video data to measure the concentration level of students.
- To provide class (attentive, inattentive & sleepy) and introduce a very light and accurate model to measure the concentration level of students.
- To introduce an effective system in the education sector to improve the learning experience of online courses.

1.3.1 Thesis Organization

The remaining sections of this paper are structured as follows:

- The literature review is presented in Chapter 2.
- The dataset collection and methodology are outlined in Chapter 3.
- The implementation is described in Chapter 4.
- The results are analyzed and discussed in Chapter 5.
- The conclusion and future work are mentioned in Chapter 6.

Chapter 2

Literature Review

The analysis of acquiring knowledge tendencies has become more important on modern online learning platforms. A new study investigates how participants interact with MOOCs (massively open online courses) and how that impacts their educational outcomes using an approach that utilizes mixed methods. Because the open online platform has online review features, participants will utilize electronic word of mouth, otherwise known as eWOM [8]. To eliminate product uncertainty and gain product knowledge, consumers are increasingly turning to the online evaluation feature. Measuring students' attentiveness is critical to effective learning, continuous improvement, and feedback. The research [3] classifies participants as "active" vs "non-active" depending on their MOOC activities throughout certain times using mobility logs from the Open EdX digital platform. It highlights some criteria, such as video widths, watching beginning times, as well as associated periods, and also acknowledges some limitations, like the difficulty of recording off-task behavior while playing back videos. The amount of time a learner may spend watching videos can be utilized in comparing their endurance to that of a mechanical part that fails. This mechanism is connected to fatigue and stress. Because it significantly compares programs with short clips to those with lengthy films, it exercises caution when offering nasty comments. Once more, synthetic measures have been added to enhance the assessment of MOOCs' attractiveness by offering a more nuanced picture of student participation in addition to course recognition, like the general h-index, g-index, and similar analogs [6].

To analyze the behaviors, two purposes are served here [7]. A survival function, which indicates the proportion of learners who keep watching a course after a predetermined amount of time, is the initial. Secondly, the hazard function describes these students' rate of dropout over the period. The hazard function of the empirical evidence is again a concave curve, but it grows thinner in the tails. The concave portion resembles a bathtub's curve, whereas the falling tail segment exhibits the Lindy effect. Likewise, it provides significant information regarding the flexible and varied students' interactions with the course material. Throughout the very first minute of watching, an improvement in enrollment rates was seen; this was explained by the power-law concept and early-reducing dropouts. Although the strategy may not be common across varied course environments, the implementation of heterogeneity in features aims to account for specific variances across participants.

Another research work [2] focuses on Shenzhen University's Fundamentals of Computing Technology program and offers a thorough explanation of the format, length, and bachelor students' approach to assessments. This study makes use of advanced analytical methods to understand and analyze human actions in MOOCs, including behavioral scenarios, clustering (k-means), and categorization (Support Vector Systems and AI Neural Networking). With the help of these models, Mumford's and Honey's methods of evaluating methodologies of learning and data are better understood, allowing participants to be classified as primarily energetic, inactive, or both simultaneously. Observations as a result of showing connections among active forum actions and educational styles. Although passive pupils are less engaged in discussions and show a desire for more difficult assignments, active learners are more engaged.

In recent decades, there has been a rapid growth in AI techniques to anticipate rates of abandonment. The most widely utilized statistic in MOOC failures is the data from the clickstream. Through the application of clickstream information as well as educational analytics to comprehend dropout habits, especially during brief periods. [14] The split of the information gathered was unbalanced, with 75.75% of learners receiving qualification and 24.25% not receiving it. The selected algorithms were determined by their effectiveness in predicting future outcomes in the context of educational data mining (EDM) and learning analytics (LA) [12]. Furthermore, a particular data set generated by the "XuetangX" online course platform is used in some research, and it is not made clear whether the suggested model can be applied to other online course platforms as a variety of student groups. Resolving this issue would improve the model's generalizability outside of the particular dataset that was utilized [10]. Here, the fast CNN strategy's ability to convert low-dimensional, depth characteristics into large-dimensional, sophisticated characteristics is a strength. For academic predictive programs, there is a strong link connecting the final success of students and their education-related behavior traits. Using strong techniques for prediction is essential to tackling the problem of discovering students who are in danger of skipping out and increasing overall curriculum performance. Another research work [5], suggested the approach involves tensoring online course data to get over these obstacles. Local tensors are used to record course-specific behaviors, and a global tensor is used to describe the entire dataset. Furthermore, the analysis of immense digital program data has gotten significantly easier with the use of leveraged natural language processing and has shown effectiveness in context assessment. the Research has examined grammatical features, such as viewpoint, to quantify language adaptability and evaluate psychological presence in conversations. NLP techniques have also been used to examine involvement sentiments and feelings in course platforms. So, the learning outcomes results (LOR) and adaptation results (AR) are both of the primary portions of the results.

In another study [3], a semi-organized approach was used to collect qualitative information with an emphasis on adoption and experiences in education. Several 771 students make up the participants, consisting of 752 students. Here, statistical procedures such as t-tests, welch t-tests, Chi-square testing, and matching of propensity scores are used in the analysis. NVivo 11 software is used to analyze qualitative data in addition to the statistical evaluation. Again in paper[12], various metrics were

used to evaluate the results in a different study, such as the Matthews Correlation Coefficient (MCC), efficiency, Area Under Curve (AUC), accuracy, recall, F1-score, and Kappa Sigma. But three models behaved better than the others: LR (logical regression), light, and GB (gradient strengthening). When it comes to identifying students who may struggle in a wide digital course, prediction models in particular—LightGBM, GB, and LR—serve as an invaluable early warning system. By Week 0, for instance, accuracy was over 83%; by Week 1, it was 93.4%; and by Week 2, it was even higher at 94.41% (excluding LDA). In each of Week 1 as well as Week 2, the F1-score, which is the harmonic median of recall and precision, showed 87.05% and 88.91% of possible points. As a result, the developed models had a significant amount of class independence, as evidenced by the AUC values, which varied from 0.9376 to 0.9863 [12].

Every learner course pair’s performance status is represented by a characteristic matrix, which is fed into the CNN. A logistical approach categorized by binary (absenteeism or persistence) is included in the architectural sixth layer, which also consists of convolutional and fully linked layers. During training [13], the corrected linear unit (Relu) activation function and root mean square as a way of prop (RM-Sprop) optimization are used. So, the outcomes of the experiment show that the suggested CNN model can accurately forecast dropout rates in MOOCs. The accuracy, recall, F measure, and preciseness are all higher with the model than with baseline techniques. Precision, recall, and F-measure metrics highlight the model’s ability to identify true dropout cases, with precision being approximately 10% better than the average precision of baseline methods. The research [15] highlights how crucial it is to keep the habits of learning in the features matrix in the correct temporal sequence because tampering with the sequence of events has a substantial impact on prediction accuracy. Again, the issue of kids’ short attention spans has been addressed with the ALIVE Minds program. A circuit board known as the ALIVE Minds Controllers (AMC) was created to track neural activity. According to preliminary data, students’ focus improved on average by 18.77% when computing and 39.45% when snoozing [4].

The relative importance of discrete behavioral data cannot be determined by CNN’s algorithms’ automatic methods for removing features. It employ the FWTS-CNN prediction method, which performed better than comparable models using the same data set, owing to a comparison investigation [10]. To overcome the drawbacks of current models, the FWTS-CNN forecasting system is presented as a solution that combines feature engineering with time series. The shortcomings in handling the fluctuating and non linear relationships found in online instructional data by typical automated learning methods. Next, it emphasizes how networks using long-short-term memory (LSTM) and recurrent neural networks, or RNN, can be used to increase accuracy in predictions, but it also underlines how these networks have limitations when it comes to capturing long-term relationships [9]. The findings indicate that the FWTS-CNN method performs better than the baseline models, proving how well it can handle the difficulties associated with MOOC dropout prediction. The models’ effectiveness is emphasized using visual aids. For the models to successfully forecast online course disengagement patterns, layers of convolutional neural networks are essential for feature extraction. It allows instructors to see pat-

terns of students' temporal interaction in online courses neatly and cleanly, so they can make informed decisions to support students in their next steps of acquisition [20].

Furthermore, it employs both facial tracking data and facial emotion detection results to figure out blend design weights for blend-style and end-style face modeling. Simple overseas pre-processing and face labels are not needed for this technology, which makes it easy to employ for regular users. In real-time expression, videos are produced via an algorithm called the First Order Action Model (FOMM). It is confirmed by experimental results that the suggested method produces natural face movies with a range of controlled attitudes [15]. Another paper [1], the video dataset and a newly created database created expressly for this study were used in the experiments, which demonstrated a high attitude categorization rate of 99.2%. In a nutshell, there are several benefits to tracking students' levels of concentration when learning online, such as warning the instructor when a significant portion of the class is unable to concentrate or providing suggestions for refocusing diverted participants.

2.1 Existing Research Gaps

It is important to disclose the selection procedure and any biases because the publication [3] lacks comprehensive information regarding the selection method used for the 771 participants, raising concerns about possible sampling bias. Also, while unstructured telephone conversations were used to collect qualitative findings, the security of the subjective findings was diminished by the lack of information regarding interview methodology and assessment techniques. Furthermore, the publications' lack of clarity in identifying elements, control procedures, and statistical analysis modifications reduces the trustworthiness and clarity of quantitative results. Research work [7] has a flaw in that data from logs (icourses) cannot be utilized to identify viewing while students are offline and a video is playing. Additionally, a few frequently used video features, such as pause, advance or reverse, and speed adjustment, are not covered in this article. In a study [15], there is not a thorough examination of the suggested frameworks' complexity. It's critical to address overfitting related problems, particularly when using models based on deep learning. The paper also talks about how different architectures work, but it doesn't address how the models can be understood. A practical use of these sophisticated models of dropout prediction requires an understanding of their workings. It will be easier to understand this crucial component of the model if intricate mathematical procedures are made simpler and given a more natural explanation. Another paper [13], is severely limited. It employs a predetermined set of five questions, which might leave out certain student issues. Here, automated techniques can increase objectivity, while manual sentiment classification creates bias. It's possible that textual sentiment doesn't always match up with user ratings when quantifying comments. Furthermore, the report does not investigate if the conclusions can be applied to other platforms or cultural contexts. As a result, we can say that there is no such paper that directly worked on students' concentration levels to analyze the problem of their poor outcomes. So we tried to fill the gap by doing our research in this area.

Chapter 3

Methodology

3.1 Proposed Methodology

Figure 3.1 illustrates the overall system for measuring students' concentration level

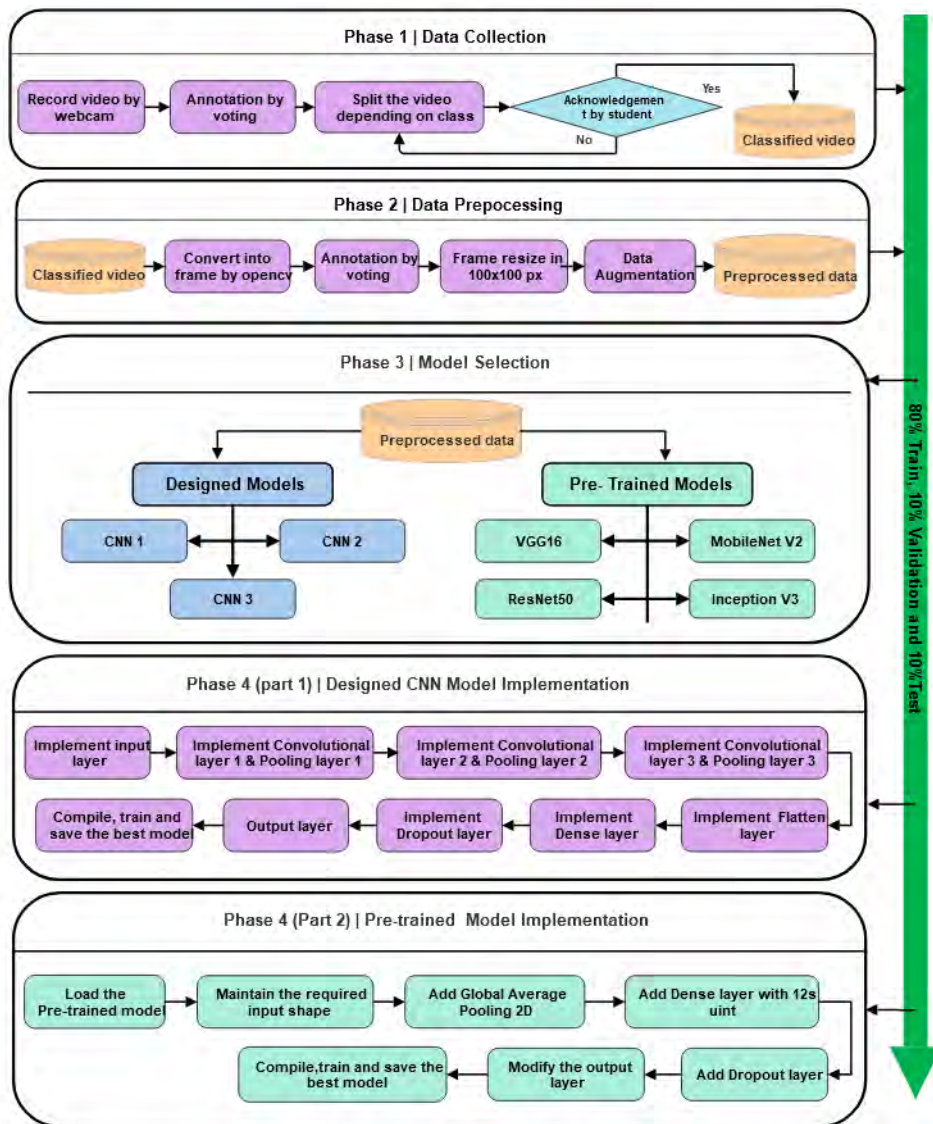


Figure 3.1: Top level overview of the method

of several phases. Raw images that we collected from the videos were first passed through the pre-processing phase. Also, data augmentation (horizontal flip, zoom (0.5)) was done in the pre-processing phase as well. The pre-processed data set was then divided into a training set, a validation set and a testing set. We trained the selected models in phase 2 using the training data. The performance of our models were measured with the following evaluation metrics: confusion matrix, precision, accuracy, recall, and F1-score.

3.2 Dataset

To understand the reason behind the dropout of students in online courses, we meticulously crafted a custom dataset for our research. We realized it was crucial to have our very own image dataset to delve deep into the matter of measuring students' concentration levels.

3.2.1 Data requirements:

Our research focuses on measuring students' concentration levels to understand the shortcomings of course materials and dropouts, but there is no specific dataset that we could use for this purpose. For this absence in the existing dataset, we needed to create a new dataset to conduct our research. This dataset includes a range of concentration levels that helped to conduct the research and contribute to advancing the fields of deep learning and image processing.

3.3 Data collection methodology

3.3.1 Recording videos

We selected a course on DataCamp and recorded ourselves while going through the course videos. Nine individuals contributed to creating a total of nine videos. Each video spanning 30 minutes. We used various devices including phones, laptops, and web cameras to create the videos.

3.3.2 Video annotation

After creating the videos we rewatched the videos and did annotations and selected 3 classes by voting. The three classes we selected were "Attentive", "Inattentive", and "Sleepy". For example, we observed that person X appeared sleepy during 5 to 7 minutes of their videos. Then we cross-checked our classification by asking X whether they were indeed sleepy or not at that time. After confirming, we filtered out the videos based on these three classifications. If they denied the classification we repeated the process from annotation again.

Symptoms	Time-duration	Class
Watch Lecture	0:00-0:40	Attentive
Didn't watch lecture	0:41-0:47	Inattentive
Watch Lecture	0:48-1:49	Attentive
Eyes closed	1:50-2:23	Sleepy
Watch Lecture	2:24-3:49	Attentive
Talking over phone	3:50-3:59	Inattentive
Watch Lecture	4:00-5:46	Attentive

Table 3.1: Preliminary video data annotation

3.3.3 Splitting and classification of the videos

After finalizing the annotation process and subsequent cross-verification, we split the videos based on the defined classifications which are “Attentive”, “Inattentive”, and “Sleepy”. Consequently, we obtained many sub-videos belonging to the three different classes. Next, we organized and segregated these videos into three separate files following the three classes. On average, each class comprised 10 videos. Thus we completed our classification.

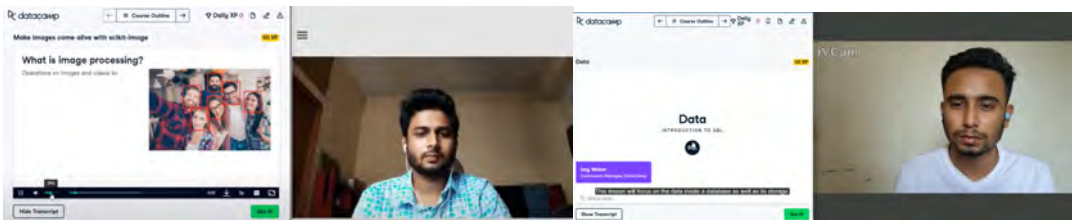


Figure 3.2: Attentive



Figure 3.3: Sleepy



Figure 3.4: Inattentive

3.4 Data pre-processing

Data pre-processing is very essential in the context of Data analysis. It improves the quality of data and transforms the data in such a way that the models can be easily read or trained with accurate data. Pre-processing also handles missing data which makes the dataset suitable for analysis. Moreover, different models have different requirements for dataset characteristics, and pre-processing meets those specifications. It ensures that the data is tailored to the different needs of each model, allowing for optimal performance of different modeling approaches.

3.4.1 Converting to frames

During this phase, we converted the videos into frames using OpenCV. OpenCV is a Python library. It helps in object detection, face recognition, image processing, etc. Typically, the videos we watch are recorded at 60 frames per second (FPS). However, for our research purposes, we specifically chose 3 frames per second (FPS). If we captured 60 frames per second it could result in redundant frames within one second. To mitigate this redundancy, we captured only the first, middle, and last frames.

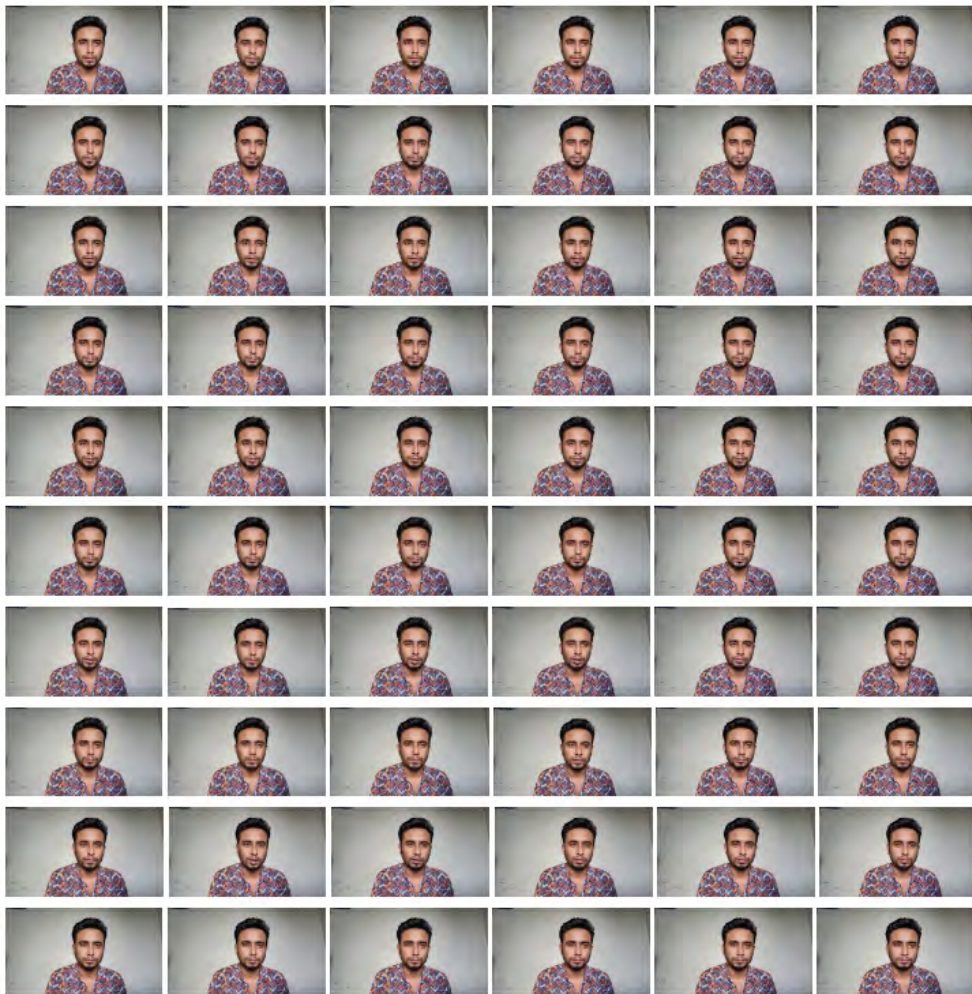


Figure 3.5: Frames in 1 second (Attentive)



Figure 3.6: First, middle and last frames

3.4.2 Frame Annotation

To avoid anomalies in the dataset we repeated the annotation process. Suppose we are converting an attentive video of person Y into frames. While doing so there is a possibility that one frame captures an image where person Y’s eyes are closed. If the machine is trained using this frame, it might incorrectly conclude that person Y was sleeping during that time which will lead to inaccurate results. Therefore, conducting the annotation again in this step is essential to ensure the accuracy of the dataset.

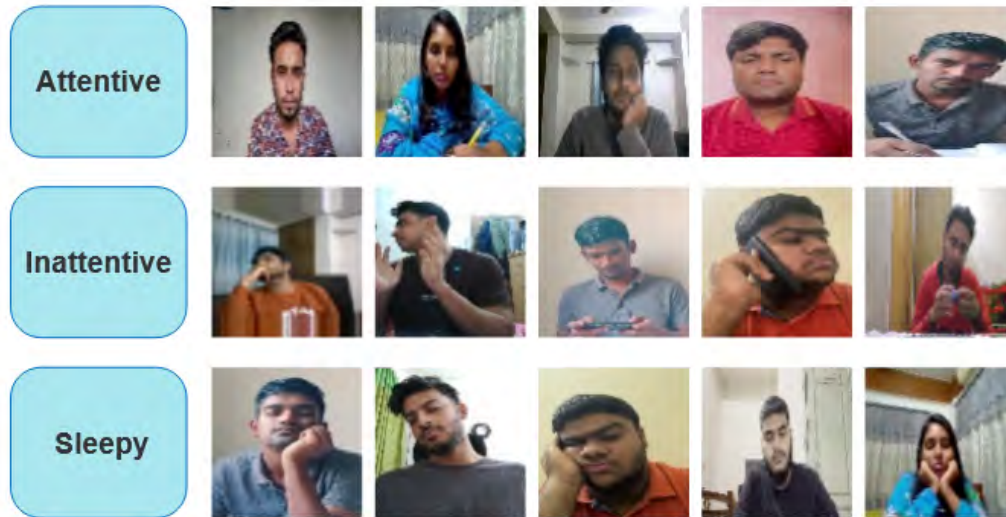


Figure 3.7: Dataset Image classification Samples

3.4.3 Resizing images

Resizing is very significant in data preprocessing as different models have distinct requirements for image input. This is pivotal for achieving optimal results. We used VGG16, MobileNetV2, ResNet50, and InceptionV3 as our pre-trained model. In VGG16, ResNet50, and MobileNetV2 image size requirement is 224*224, and in InceptionV3 229*229. To meet these criteria, we resized our images to 100*100 to make sure they fit the specifications of all the models.

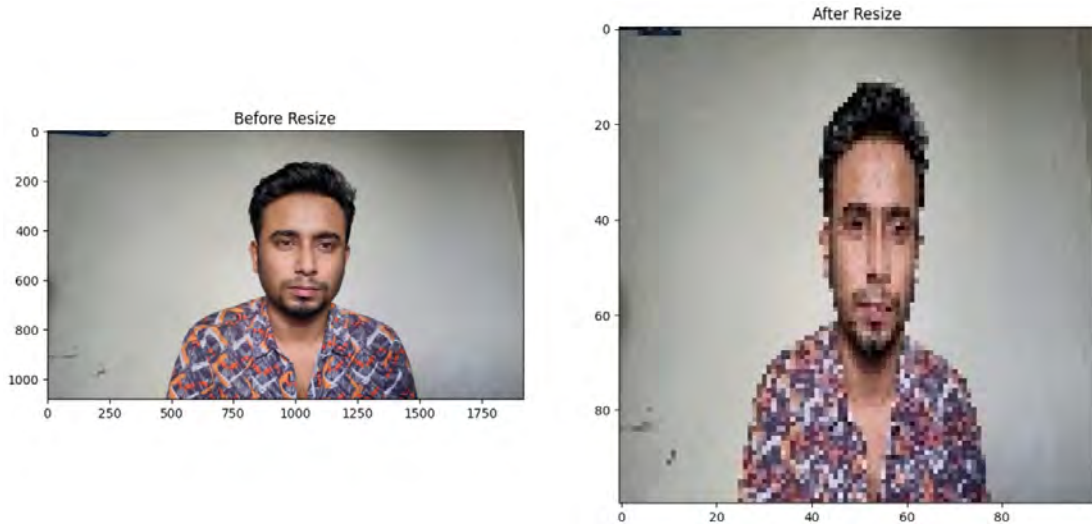


Figure 3.8: Before and after resize

3.4.4 Data Augmentation

Data augmentation is a machine learning technique. It expands a dataset by applying various transformations. Thus, it increases the diversity and variability of the data. There are some common techniques for data augmentation: rotations, flips, zooms, and brightness changes. It prevents overfitting, increases accuracy, and reduces the operational cost of the models. In image processing, it increases performance and resilience to handle real-life scenarios. For our dataset, we used the horizontal flip and zooming technique. The zoom range is 0.5. As our dataset comprises images of people, we didn't apply rotation, stretching, or cropping techniques because these wouldn't be meaningful in this context.

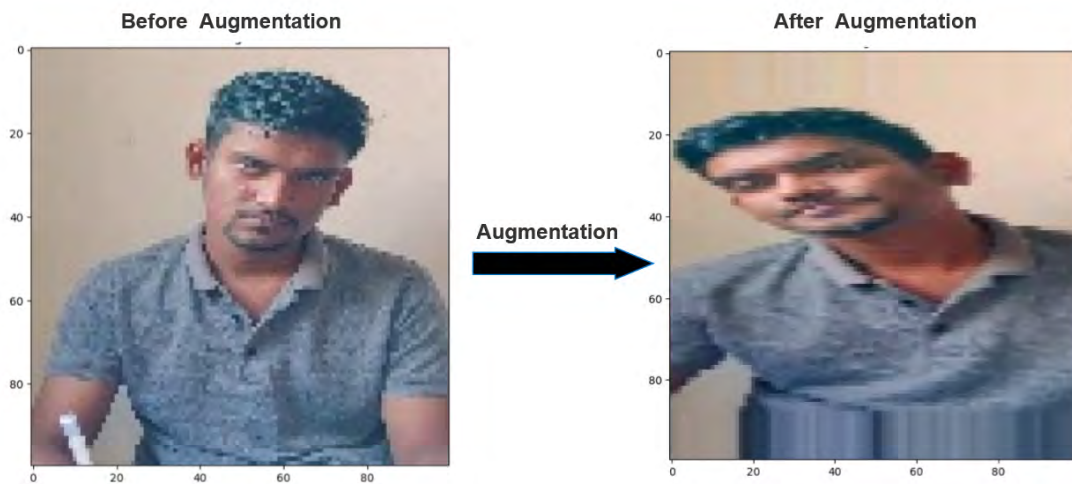


Figure 3.9: Before and after augmentation

3.4.5 Rescaling

Scaling is a fundamental operation in image processing. Input values within a specific range enhance the stability of Neural network performance. A common way to normalize a picture is to divide the pixel values by 255 which will scale down the picture to the range [0,1]. Normalizing images ensures that all images in the dataset are on a consistent scale. It helps in handling computational resources efficiently. Moreover, small images reduce computational load and this makes the process faster.

We have images in the size of RGB in the color model, where the pixel values range from 0 to 255. For normalizing the data, we divided the pixel values by 255 and scaled down the images to a new range of [0,1]. This normalization technique was employed to maintain a consistent scale for all pixel values and enhance the model performance.

```
[[[153 147 135]
   [162 156 144]
   [169 163 151]]]

[[155 149 137]
 [164 158 146]
 [170 164 152]]

[[155 149 137]
 [164 158 146]
 [171 165 153]]]
```

Figure 3.10: Image array before rescaling

```
[[[0.6      0.57647059 0.52941176]
   [0.63529412 0.61176471 0.56470588]
   [0.6627451  0.63921569 0.59215686]]]

[[0.60784314 0.58431373 0.5372549 ]
 [0.64313725 0.61960784 0.57254902]
 [0.66666667 0.64313725 0.59607843]]]

[[0.60784314 0.58431373 0.5372549 ]
 [0.64313725 0.61960784 0.57254902]
 [0.67058824 0.64705882 0.6       ]]]]
```

Figure 3.11: Image array after rescaling

3.4.6 Pre-processed data

After implementing all the above techniques on the raw inputs, we successfully pre-processed our data, preparing it for effective model training and reliable performance. We used 80% of our data for validation, 10% for test and 10% for train data.

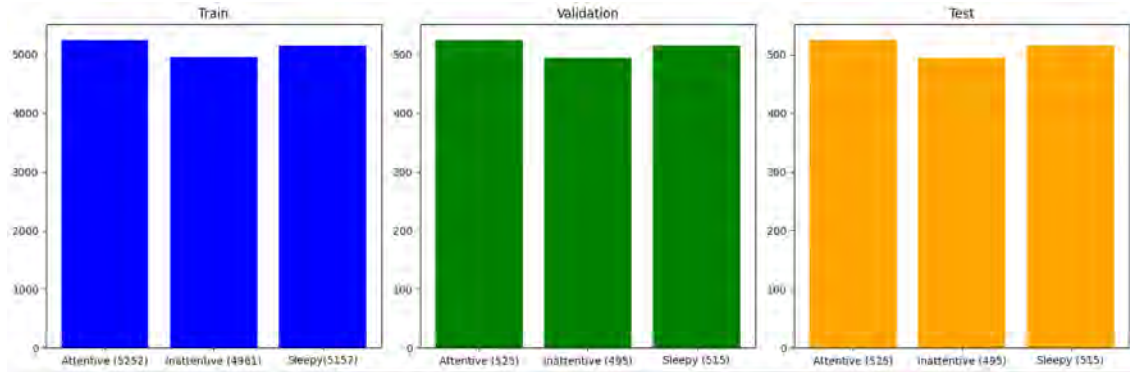


Figure 3.12: Data representation

3.5 Model Specification

3.5.1 Convolutional Neural Network(CNN)

Convolutional Neural Networks (CNNs) are a class of deep neural networks that are particularly effective for image recognition and computer vision tasks. CNN architecture typically has three layers: convolutional layer, pooling layer, and fully connected layer.

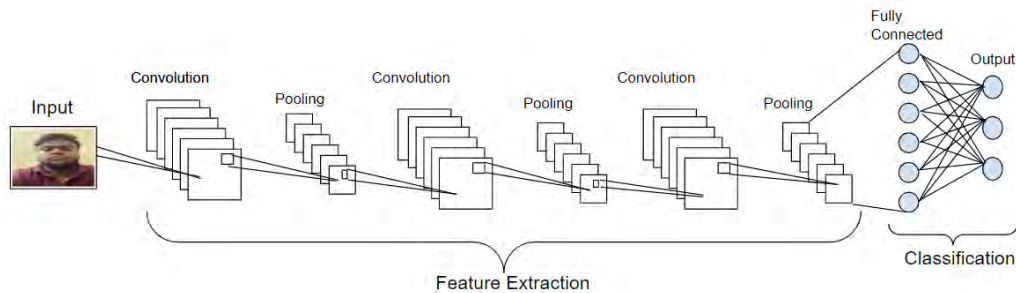


Figure 3.13: Architecture of our proposed CNN model

In convolution layer first layer is the input layer. It accepts the raw images. A digital image contains a series of pixels where each pixel has a value that represents the color or brightness of that pixel[16].

Convolution layer:

The convolution layer is the building block of CNN architecture. This is the main layer for extracting features from raw input images. This layer performs a dot product between the input image and a kernel or filter of a particular size. The kernel size is spatially smaller than the input image. To identify patterns such as edges,

textures, and more intricate structures, these filters glide across the input. The kernel's sliding size is called a stride.

Suppose a black-and-white image size is $n * n * 1$ and the filter size is $f * f * c$ then the formula:

$$(n - f + 1) * (n - f + 1) * c \dots \dots \dots (1)$$

For color image (RGB), the image size is $n * n * 3$ and the kernel size is $f * f * 3$ then the formula:

$$(n - f + 1) * (n - f + 1) * 1 \dots \dots \dots (2)$$

Here, n = image size, f = filters, c = number of filters.

The output of this layer is the featured map. It contains information about the input image such as the corners and the edges. Then this map is given to the next layer to extract more features and information of the image.

Pooling layer

Following a Convolutional Layer, a Pooling Layer is added to diminish computational costs by reducing the dimensions of the convolved feature map. Three kinds of pooling are there: max pooling, average pooling, and sum pooling. In max pooling the largest value is taken from the feature map for further operation. In average pooling average value of the feature map is taken. In sum-pooling sum of all the feature map values is taken.

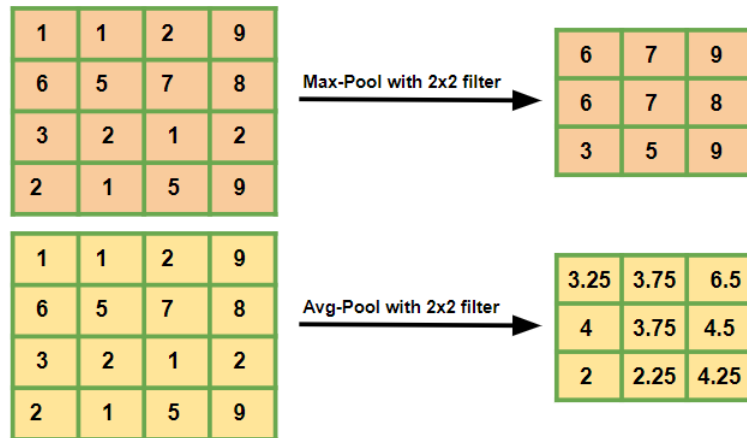


Figure 3.14: Example pooling

The pooling layer generalizes the features that are extracted from the convolution layer or the feature map and also diminishes computations within the network. Thus it works like a bridge between the convolution layer and FC layer.

Fully connected layers:

In fully connected layers, every neuron is connected with every neuron of the next layer. In this step, the input image we've been working with gets flattened and fed to the Fully Connected (FC) layer[17]. The flattened layer goes through a few more FC layers and the classification process begins to take place. Connecting two layers is better because two FC layers work better than just one.

3.5.2 MobileNet V2

MobileNetV2 model has 53 convolution layers. It is a type of convolutional neural network architecture designed specifically for mobile and edge devices. It has one AvgPool with nearly 350 GFLOP. MobileNetV2 uses depthwise separable convolutions to factorize the ordinary convolution into depthwise and pointwise convolutions[11]. Because each input channel has its spatial filter applied individually, this architecture lowers computing loads. For real-time image processing jobs and on-device machine learning applications, its careful trade-off between computing efficiency and model performance makes it a very valuable architecture. There are two blocks:

- Inverted Residual block with stride 1.
- Bottleneck Residual Block with stride 2.

Internal components of stride 1 and 2 blocks:

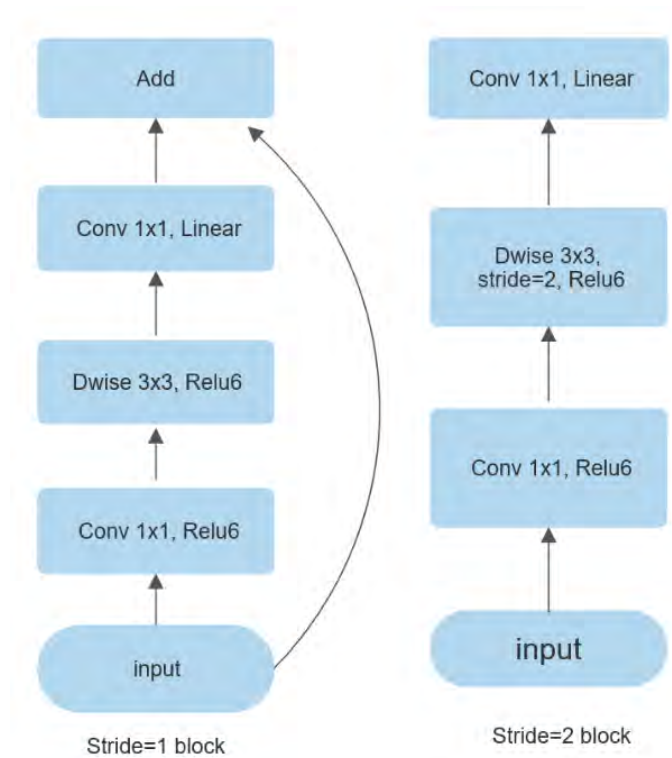


Figure 3.15: Fully connected layer

There are three different layers for each block:

- 1x1 Convolution with Relu6
- Depthwise Convolution
- 1x1 Convolution without any non-linearity

3.5.3 VGG 16

VGG16 is a deep convolutional neural network architecture that is considered to be one of the best computer vision models to date. It is one of the popular algorithms for image classification. VGG16 architecture has 16 weight layers, 13 convolutional layers, and 3 fully connected layers[19]. The 16 in VGG16 denotes 16 layers that have weights. It uses small 3x3 convolutional filters with stride 1 and the same padding and max pool layer of the 2*2 filter of stride 2 for finer spatial details. As a result, it showed a significant improvement on the prior-art configurations. Later this architecture uses max pooling to reduce spatial dimension. VGG16 consists of three fully connected layers with 4096 neurons, followed by a final output layer with the number of neurons needed for the classification task. VGG16 was created to handle 224 by 224-pixel input pictures with three RGB color channels.

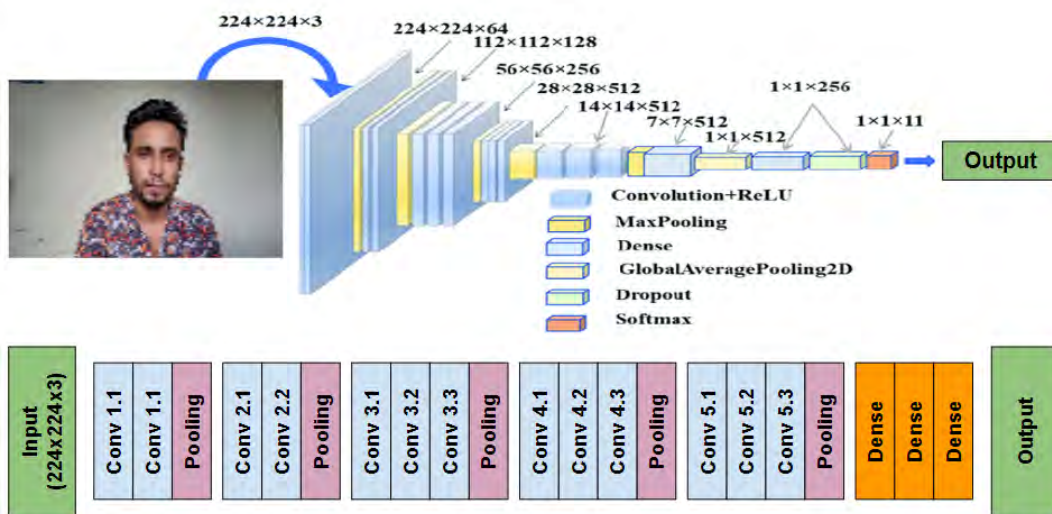


Figure 3.16: VGG16 architecture

3.5.4 ResNet50

ResNet50 is a powerful classification model. The ResNet50 architecture can be divided into four main parts. They are the convolutional layers, the identity block, the convolutional block, and the fully connected layers. The convolution layer extracts features from the input image, identity, and convolution block process and transforms these features and a fully connected layer makes the final classification. The convolution layers of ResNet50 consist of several convolution layers[21]. These layers extract the features such as edges, textures, and shapes from the input image. After these convolution layers, there are max pooling layers that decrease the spatial dimension of the feature maps. The identity block helps the network learn residual functions that map the input to the desired output. In the convolutional block, the 1*1 convolutional layer is used to reduce the number of filters before the 3*3 convolutional layer. Fully connected layers indicate the final classification and the output is given to softmax activation function.

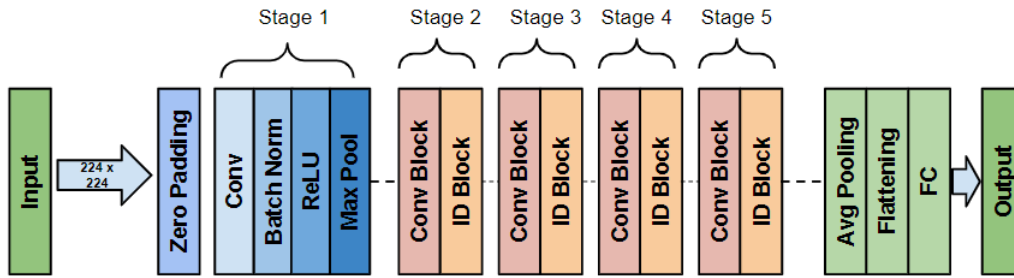


Figure 3.17: ResNet50 model architecture

3.5.5 InceptionV3

InceptionV3 was developed by Google. It is a very powerful architecture for image processing and classification. It uses 1×1 convolutions to improve computational efficiency. It accelerates training and enhances robustness of the model by batch normalization. It also has auxiliary classifiers. Overall, InceptionV3 marks a milestone in the convolutional neural networks world. It demonstrates a brilliant design that balances among complexity, efficiency and high-performance in various visual recognition tasks[18].

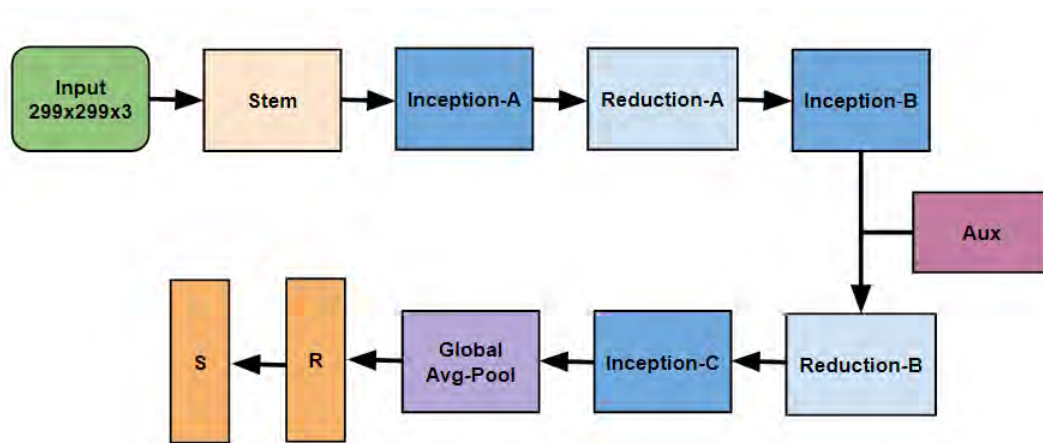


Figure 3.18: InceptionV3 architecture

Chapter 4

Implementation

4.1 Workflow

Our research journey started with the creation of a unique dataset. Given the absence of an available dataset, we recorded 30-minute videos from 9 individuals to evaluate concentration levels as a key factor in understanding student dropouts. As a result, we followed a long process to pre-process our dataset which also involved dividing the dataset into three segments: 80% for validation, 10% for training, and another 10% for testing.

Moving forward, the model selection process began, where we identified effective models for training in concentration level measurement. Our chosen models fell into two categories: 1) Design models, including ConcentrateNet1 and ConcentrateNet2, and ConcentrateNet3) Pre-trained models such as VGG16, MobileNetV2, InceptionV3, and ResNet50.

With the models selected, we proceeded to set them up, execute them, and fine-tune hyperparameters. The implementation phase encompassed both design models and pre-trained models.

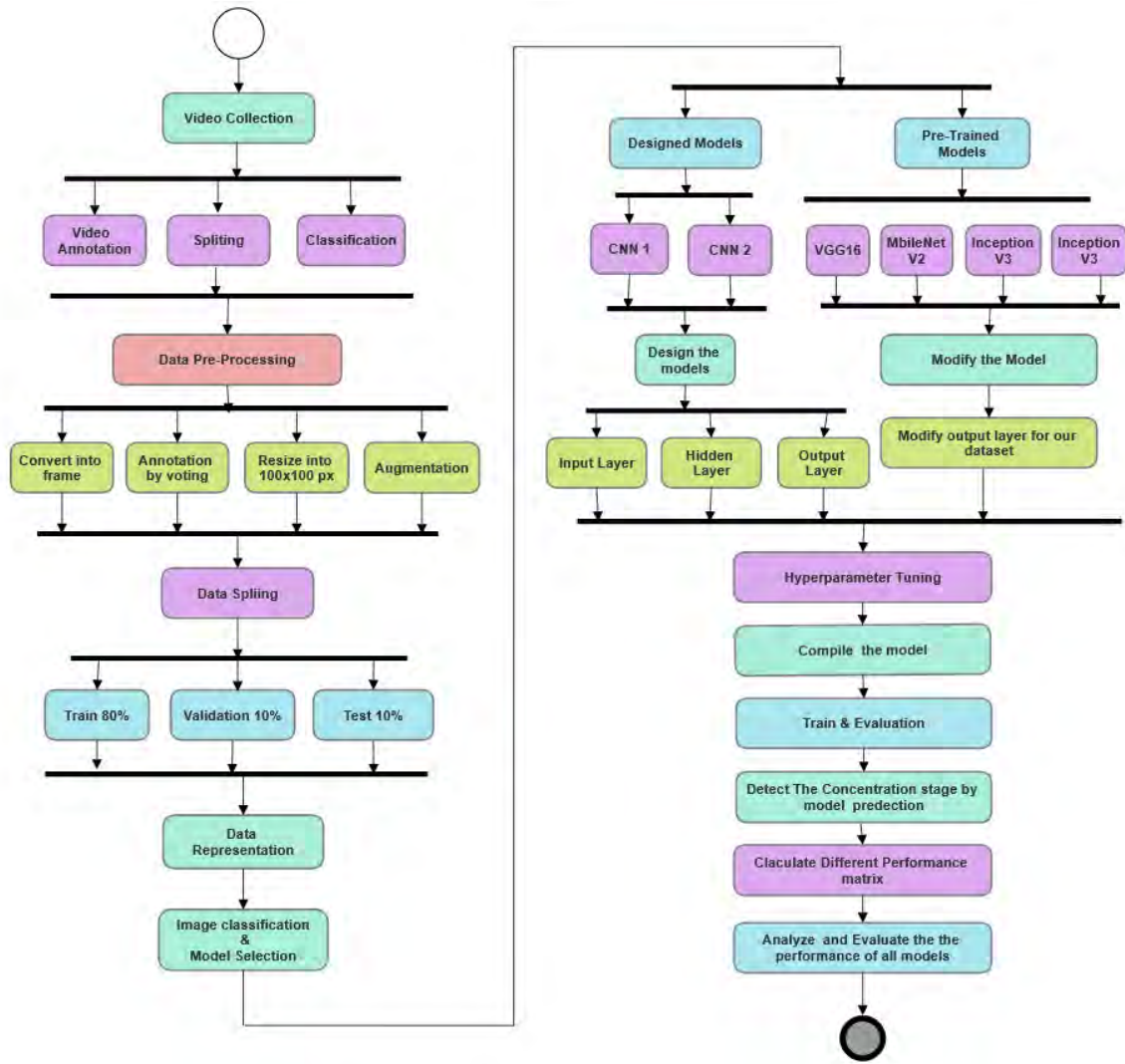


Figure 4.1: Workflow details for implementation.

4.2 Setup for Experiment

4.2.1 Training Hardware and Software

Experiments have been carried out on a device with the hardware characteristics demonstrated below.

Table 4.1: Summary of hardware characteristics

CPU	Core i7 8th Gen
GPU	NVIDIA GeForce GTX 1050 Ti
RAM	12GB
ROM	1TB
OS	Windows 11

On this device we used Google Colab for running the code.

4.2.2 Library List

Table 4.2: Summary of Python library list

Serial No	Library	Version
1	OpenCV	4.8.0
2	Numpy	1.23.5
3	Matplotlib	3.7.1
4	Pandas	1.5.3
5	SKLearn	1.2.2
6	PIL	9.4.0
7	Keras	2.15.0
8	Tensorflow	2.15.0

4.2.3 Structural view of the code skeleton

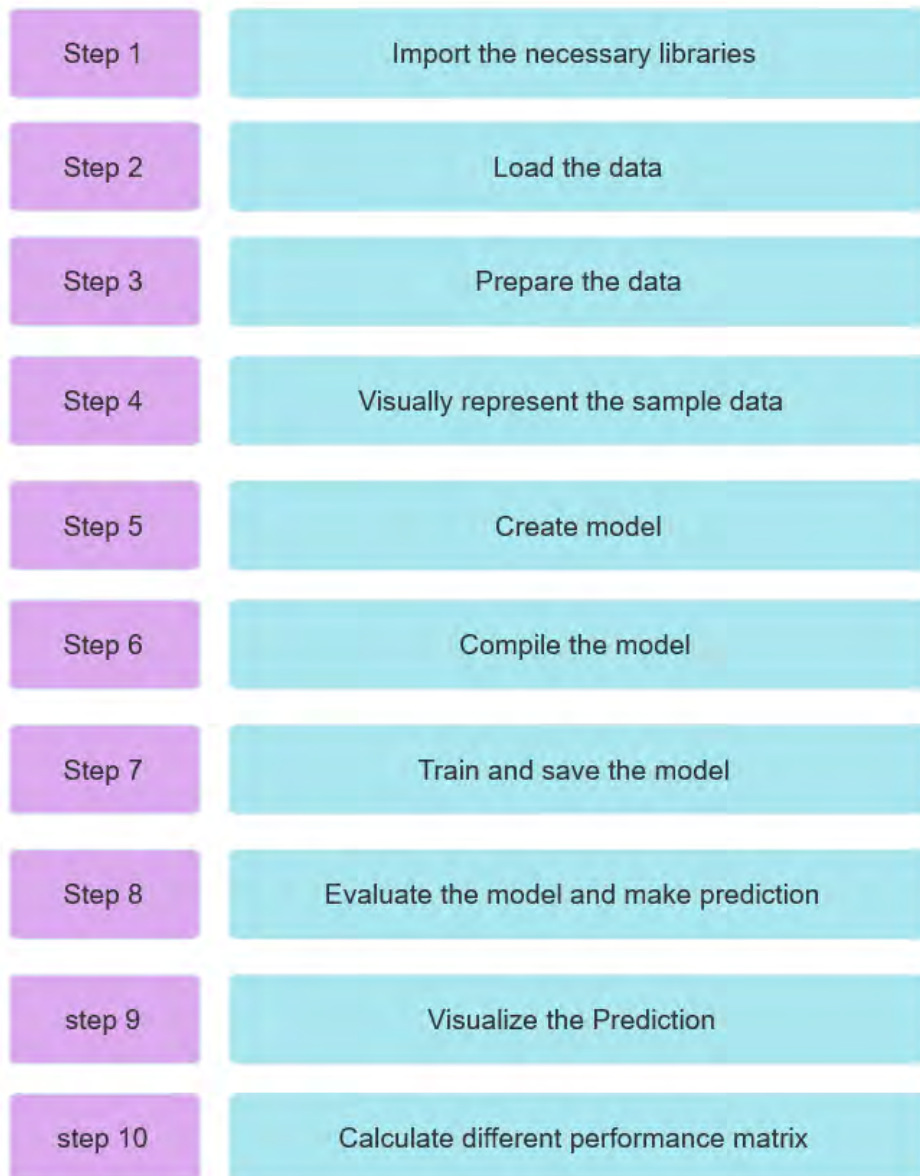


Figure 4.2: Structural view of the code skeleton

4.3 Pre-processing & Augmentation

4.3.1 Pre-processing

Image data is generated from the classified videos. The image data is converted by generating frames from video using the OpenCV library. Usually, there are 40-60 frames in 1 second video. But we took only 3 frames from every one second. If we think about 1 second then we take the first, middle, and last frame. Because, in 1 second video there are multiple same frames. Which creates redundancy in the dataset. Next, we check every frame whether the frame is in the correct class or not. The classes are showing in Figure 4.3 Our next goal is to reduce the size of the

dataset. Since we have almost 15 thousand images so if we do not reduce the size of the images then the dataset will be huge. Moreover, it will be very difficult to work with this large sized dataset. That is why we resize every image to 100x100. It will also help to ensure consistency in the dataset. The images are both in PNG and JPG format.

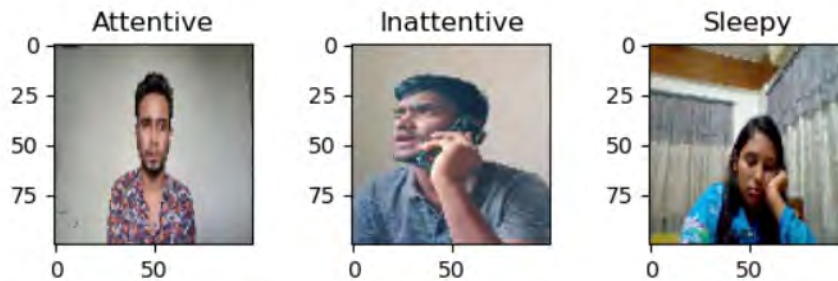


Figure 4.3: Image classification

4.3.2 Augmentation

The augmentation techniques help to increase the diversity of the dataset. There are different types of augmentation techniques such as flip, rotation, shearing, zooming, etc. We augment the dataset with ImageDataGenerator by giving two arguments. The arguments we are giving in the parameter are shown in the Table 4.3. The first argument is horizontal flip. That means we wanted to create a mirror image of the original image then zoom the image. The zoom range was 0.5. That means we take an image and augment the image by horizontal flipping and zooming Figure 4.4 Then we rescale the image. That means we normalize the pixel value by dividing

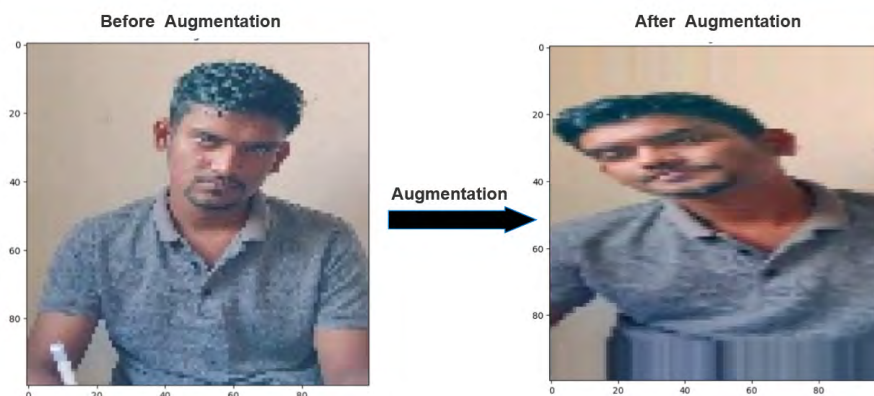


Figure 4.4: Before and after augmentation

255. As a result, the range of pixel value of any image should be in between 0 and 1. This helps the model converge faster during training.

Table 4.3: Parameters used in data augmentation

Generator Type	Facility
Horizontal Flip	True
Zoom	0.5 (range)

Table 4.4: Number of samples before & after augmentation

Classes	Samples before augmentation	Samples after augmentation
Attentive	2797	5252
Inattentive	2634	4961
Sleepy	2768	5157

4.4 Model selection

To train the image dataset for measuring the concentration we try to select lightweight models to classify the images. To classify the images we use CNN architecture. We design 3 models based on CNN architecture. These are ConcentrateNet1(CNN1), ConcentrateNet2(CNN2) and ConcentrateNet3(CNN3). These models are very lightweight with high accuracy. Moreover, we also select 4 pre-trained models which are comparatively lighter than other pre-trained models. When it came to model selection, we

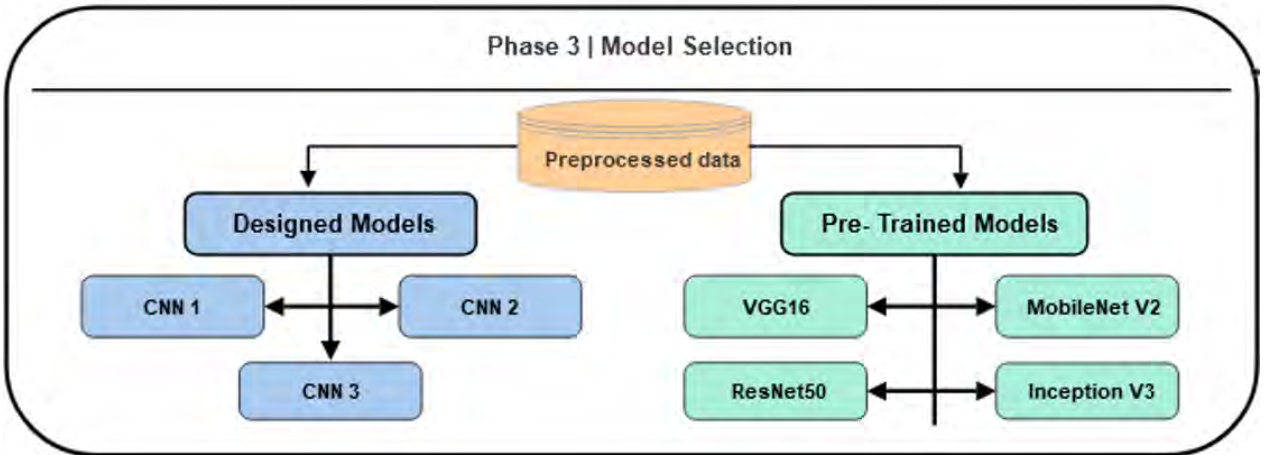


Figure 4.5: Diagram of model selection

gave a strong emphasis on both robustness and uniqueness. We carefully handle the transfer learning layer. Because these models have never seen these data previously.

4.5 Hyperparameter Tuning

Hyperparameters are the parameters that we pass when a function or class being called. The parameter value is very important when it comes to the performance of an algorithm. Perfect parameter value boost the performance of an algorithm. The

same goes for a model. It has a direct impact on the model's performance. Choosing the proper hyperparameter values is a crucial part when it comes to the effectiveness of any machine learning algorithm. In our scenario, when we trained our inception v3 model we used Adam as the optimizer and learning rate=0.001. As a result, we got very low validation accuracy. Then we changed the learning rate from 0.001 to 0.0001 which spiked up the validation accuracy from 35% to almost 90%. In another scenario when we were trying to train the model we set the `step_per_epoc` value randomly. As a result, our model was unable to fully train because this stage our model could only learn from the $(\text{step_per_epoc} * \text{batch_size})$ images. However, we had to train the model with all of our images so we set the `step_per_epoc = number of images in train dataset / batch_size`. After that, we got our expected validation accuracy.

4.6 Design & Compile the Models

4.6.1 ConcentrateNet1

The model consists of three convolutional layers, along with a max-pooling layer with a 2x2 pool size in each convolutional layer. Max-pooling layer used for down-sampling. It reduce the image information. As a result, the computational cost get reduced. The first layer has 16 filters, the second has 32 filters, and the third has 64 filters. We used ReLU as activation function in every convolutional layer. Then we implement flatten layer. The flattened output is passed through a densely connected layer with 512 neurons and a ReLU activation function. After that, we used a dropout layer with a dropout rate of 0.2. This layer helps to allay from overfitting. Finally, we implement the output layer that consists three neurons with a softmax activation function. The softmax function is suitable for multiclass. The aim of this model is to learn hierarchical features through convolution and pooling operations, dense layers for classification, while dropout helps enhance generalization by preventing overfitting during training.

Then we compile the model using the Adam optimizer with a learning rate of 0.001. We choose categorical cross-entropy as loss function, which is suitable for multi-class classification. Additionally, the accuracy metric is specified for monitoring the model's performance during training and evaluation.

4.6.2 ConcentrateNet2

The model consists of three convolutional layers, along with a max-pooling layer with a 2x2 pool size and a batch-normalization layer in each convolutional layer. Max-pooling layer used for down-sampling. It reduce the image information. As a result, the computational cost get reduced. The first layer has 16 filters, the second has 32 filters, and the third has 64 filters. We used ReLU as activation function in every convolutional layer. Then we implemente flatten layer. The flattened output is passed through a densely connected layer with 128 neurons and a ReLU activation function. After that, we used a dropout layer with a dropout rate of 0.2. This layer helps to allay from overfitting. Finally, we implement the output layer that consists three neurons with a softmax activation function. The softmax function is suitable

for multiclass. The aim of this model is to learn hierarchical features through convolution and pooling operations, dense layers for classification, while dropout helps enhance generalization by preventing overfitting during training.

Then we compile the model using the RMSprop optimizer with a learning rate of 0.001. We choose categorical cross-entropy as loss function, which is suitable for multi-class classification. Additionally, the accuracy metric is specified for monitoring the model's performance during training and evaluation.

4.6.3 ConcentrateNet3

The model consists of three convolutional layers, along with a batch-normalization layer and a avg-pooling layer with a 2x2 pool size in each convolutional layers. Avg-pooling layer used for down-sampling. It reduce the image information. As a result, the computational cost get reduced. The first layer has 16 filters, the second has 32 filters, and the third has 64 filters. We used ReLU as activation function in every convolutional layer. Then we implement flatten layer. The flattened output is passed through a densely connected layer with 512 neurons and a ReLU activation function. After that, we used a dropout layer with a dropout rate of 0.2. This layer helps to allay from overfitting. Finally, we implement the output layer that consists three neurons with a softmax activation function. The softmax function is suitable for multiclass. The aim of this model is to learn hierarchical features through convolution and pooling operations, dense layers for classification, while dropout helps enhance generalization by preventing overfitting during training.

Then we compile the model using the Adam optimizer with a learning rate of 0.001. We choose categorical cross-entropy as loss function, which is suitable for multi-class classification. Additionally, the accuracy metric is specified for monitoring the model's performance during training and evaluation.

4.6.4 VGG16

The VGG16 is a pre-trained model which is trained on the large dataset. First, we load the model. Then, we exclude the original fully connected layers of VGG16. After that, we modify the model by adding a global average pooling layer to the output of the VGG16 base model. However, we implement a densely connected layer with 512 neurons and use ReLU as an activation function. The final output layer is a dense layer with three neurons. We used softmax activation function in output layer. Which is suitable for a multi-class classification. The resulting model takes advantage of the pre-trained VGG16 features for effective feature extraction from images.

Then we compile the model using the Adam optimizer with a learning rate of 0.001. We choose categorical cross-entropy as a loss function during compilation. which is suitable for multi-class classification. Lastly, the accuracy metric is specified for monitoring the model's performance during training and evaluation.

4.6.5 ResNet50

The ResNet50 is a pre-trained model which is trained on the large dataset. First, we load the model. Then, we exclude the original fully connected layers of ResNet50. After that, we modify the model by adding a global average pooling layer to the output of the ResNet50 base model. However, we implement a densely connected layer with 128 neurons and use ReLU as an activation function. The final output layer is a dense layer with three neurons. We used softmax activation function in output layer. Which is suitable for a multi-class classification. The resulting model takes advantage of the pre-trained ResNet50 features for effective feature extraction from images.

Then we compile the model using the Adam optimizer with a learning rate of 0.001. We choose categorical cross-entropy as a loss function during compilation. which is suitable for multi-class classification. Lastly, the accuracy metric is specified for monitoring the model's performance during training and evaluation.

4.6.6 InceptionV3

The InceptionV3 is a pre-trained model which is trained on the large dataset. First, we load the model. Then, we exclude the original fully connected layers of InceptionV3. After that, we modify the model by adding a global average pooling layer to the output of the InceptionV3 base model. However, we implement a densely connected layer with 128 neurons and use ReLU as an activation function. The final output layer is a dense layer with three neurons. We used the softmax activation function in the output layer. Which is suitable for a multi-class classification. The resulting model takes advantage of the pre-trained InceptionV3 features for effective feature extraction from images.

Then we compile the model using the Adam optimizer with a learning rate of 0.0001. We choose categorical cross-entropy as a loss function during compilation. Which is suitable for multi-class classification. Lastly, the accuracy metric is specified for monitoring the model's performance during training and evaluation.

4.6.7 MobileNetV2

The MobileNetV2 is a pre-trained model which is trained on the large dataset. First, we load the model. Then, we exclude the original fully connected layers of MobileNetV2. After that, we modify the model by adding a global average pooling layer to the output of the MobileNetV2 base model. However, we implement a densely connected layer with 128 neurons and use ReLU as an activation function. The final output layer is a dense layer with three neurons. We used the softmax activation function in the output layer. Which is suitable for a multi-class classification. The resulting model takes advantage of the pre-trained InceptionV3 features for effective feature extraction from images.

Then we compile the model using the Adam optimizer with a learning rate of 0.001. We choose categorical cross-entropy as a loss function during compilation. which

is suitable for multi-class classification. Lastly, the accuracy metric is specified for monitoring the model’s performance during training and evaluation.

4.7 Train & Evaluate the Models

4.7.1 ConcentrateNet1

We train this model by providing the train and validation dataset. The batch size is 4. So, steps per epoch = number of training samples / batch size. Which means that the training process will iterate a number of steps that is equal to one-fourth of the total number of samples in the training dataset during each epoch. The training will be executed for 10 epochs as we set epochs = 10. We also use callbacks. Which will monitor the validation accuracy and save the model which is responsible for maximum validation accuracy. We also store the training history in a variable and visualize it on a graph Figure 4.6. The figure represents two graphs one is training_accuracy Vs validation_accuracy and the other one is training_loss Vs validation_loss.

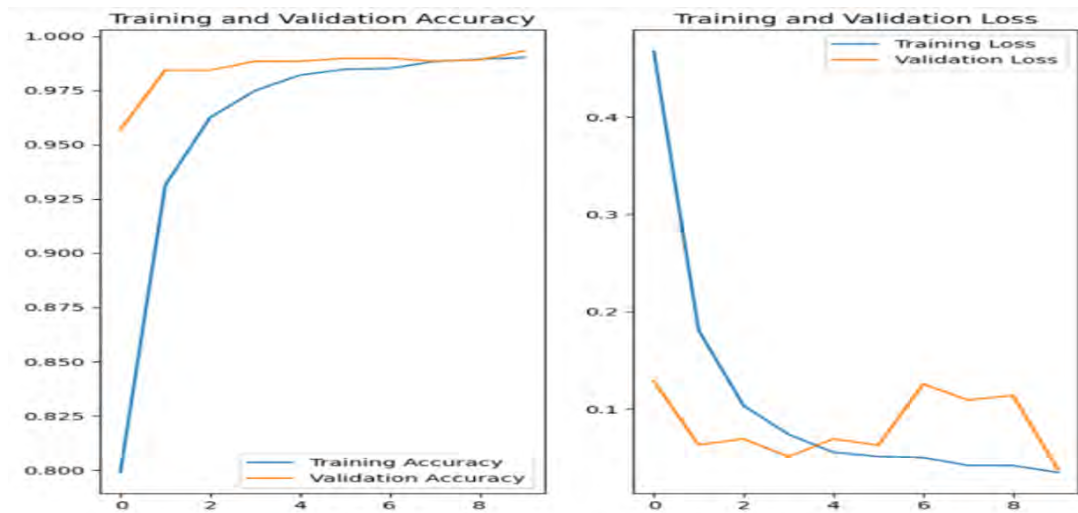


Figure 4.6: Training_accuracy Vs Validation_accuracy & Training_loss Vs Validation_loss

Then we evaluate the model with the test dataset. The performance matrices of evaluation are given below Table 4.5.

Table 4.5: Summary of performance matrices of evaluation

Metric	Value
Accuracy	0.9309
Loss	0.8629

4.7.2 ConcentrateNet2

We train this model by providing the train and validation dataset. The batch size is 32. So, steps per epoch = number of training samples / 32. which means that the training process will iterate a number of steps that is equal to one-thirty seconds of the total number of samples in the training dataset during each epoch. The training will be executed for 20 epochs as we set epochs = 20. We also use callbacks. Which will monitor the validation accuracy and save the model which is responsible for maximum validation accuracy. We also store the training history in a variable and visualize it on a graph Figure 4.7. The figure represents two graphs one is training_accuracy Vs validation_accuracy and the other one is training_loss Vs validation_loss.

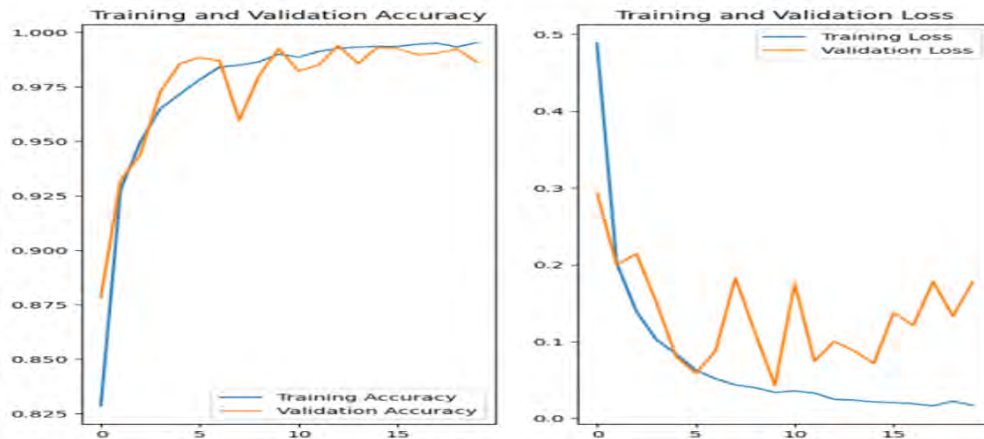


Figure 4.7: Training_accuracy Vs Validation_accuracy & Training_loss Vs Validation_loss

Then we evaluate the model with the test dataset. The performance metrics of evaluation are given below Table 4.6

Table 4.6: Summary of performance matrices of evaluation

Metric	Value
Accuracy	0.9270
Loss	1.4355

4.7.3 ConcentrateNet3

We train this model by providing the train and validation dataset. The batch size is 4. So, steps per epoch = number of training samples / batch size. which means that the training process will iterate several steps that are equal to one-fourth of the total number of samples in the training dataset during each epoch. The training will be executed for 15 epochs as we set epochs = 15. We also use callbacks. Which will monitor the validation accuracy and save the model which is responsible for maximum validation accuracy. We also store the training history in a variable and visualize it on a graph Figure 4.8. The figure represents two graphs one is training_accuracy Vs validation_accuracy and the other one is training_loss Vs validation_loss.

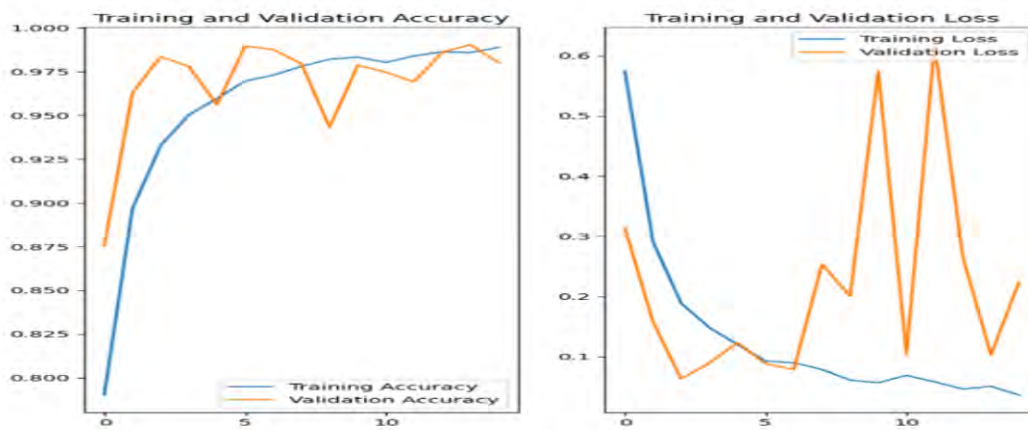


Figure 4.8: Training_accuracy Vs Validation_accuracy & Training_loss Vs Validation_loss

Then we evaluate the model with the test dataset. The performance metrics of evaluation are given below Table 4.7

Table 4.7: Summary of performance matrix of evaluation

Metric	Value
Accuracy	0.9218
Loss	2.9780

4.7.4 VGG16

We train this model by providing the train and validation dataset. The batch size is 4. So, steps per epoch = number of training samples / batch size. Which means that the training process will iterate several steps that is equal to one-third of the total number of samples in the training dataset during each epoch. The training will be executed for 10 epochs as we set epochs = 10. We also use callbacks. Which will monitor the validation accuracy and save the model which is responsible for maximum validation accuracy. We also store the training history in a variable and visualize it on a graph Figure 4.9. The figure represents two graphs one is training_accuracy Vs validation_accuracy and the other one is training_loss Vs validation_loss.

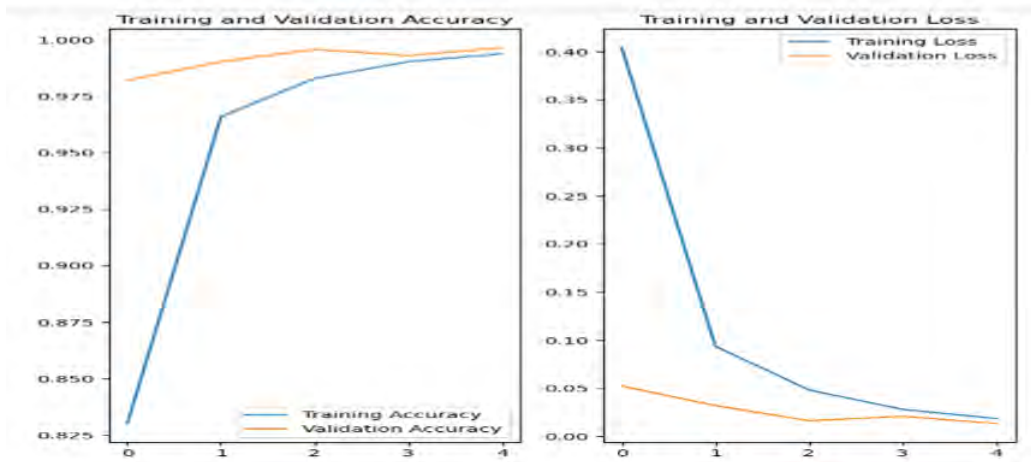


Figure 4.9: Training_accuracy Vs Validation_accuracy & Training_loss Vs Validation_loss

Then we evaluate the model with the test dataset. The performance metrics of evaluation are given below Table 4.8

Table 4.8: Summary of performance matrices of evaluation

Metric	Value
Accuracy	0.8951
Loss	1.2891

4.7.5 MobileNetV2

We train this model by providing the train and validation dataset. The batch size is 4. So, steps per epoch = number of training samples / batch size. Which means that the training process will iterate several steps that is equal to one-fourth of the total number of samples in the training dataset during each epoch. The training will be executed for 5 epochs as we set epochs = 5. We also use callbacks. Which will monitor the validation accuracy and save the model which is responsible for maximum validation accuracy. We also store the training history in a variable and visualize it on a graph Figure 4.10. The figure represents two graphs one is training_accuracy Vs validation_accuracy and the other one is training_loss Vs validation_loss.

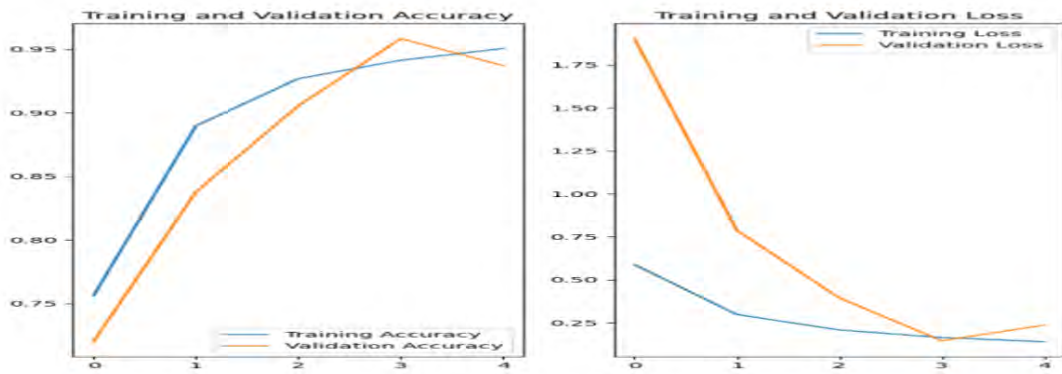


Figure 4.10: Training_accuracy Vs Validation_accuracy & Training_loss Vs Validation_loss

Then we evaluate the model with the test dataset. The performance matrices of evaluation is given below Figure 4.10.

Table 4.9: Summary of performance matrices of evaluation

Metric	Value
Accuracy	0.8579
Loss	0.6057

4.7.6 ResNet50

We train this model by providing the train and validation dataset. The batch size is 4. So, steps per epoch = number of training samples / batch size. Which means that the training process will iterate several steps that is equal to one-fourth of the total number of samples in the training dataset during each epoch. The training will be executed for 10 epochs as we set epochs = 10. We also use callbacks. Which will monitor the validation accuracy and save the model which is responsible for maximum validation accuracy. We also store the training history in a variable and visualize it on a graph Figure 4.11. The figure represents two graphs one is training_accuracy Vs validation_accuracy and the other one is training_loss Vs validation_loss.

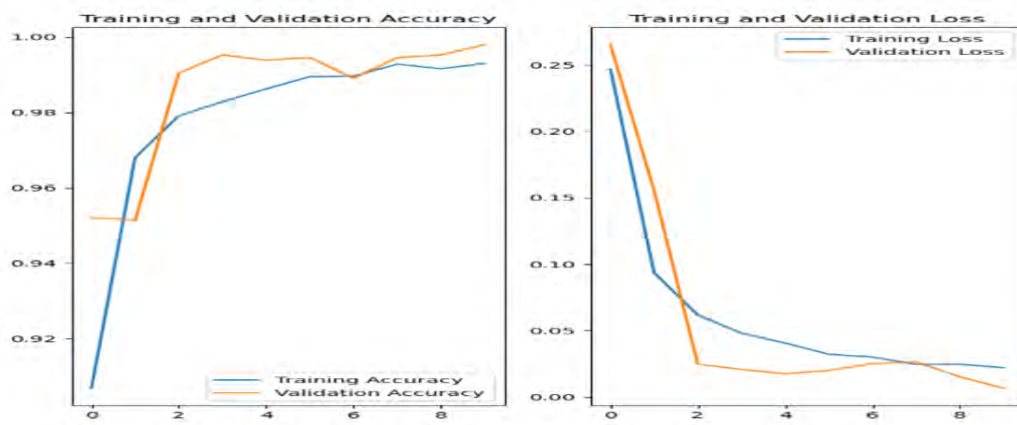


Figure 4.11: Training_accuracy Vs Validation_accuracy & Training_loss Vs Validation_loss

Then we evaluate the model with the test dataset. The performance metrics of evaluation are given below Table 4.10.

Table 4.10: Summary of performance matrices of evaluation

Metric	Value
Accuracy	0.9092
Loss	1.6781

4.7.7 InceptionV3

We train this model by providing the train and validation dataset. The batch size is 4. So, steps per epoch = number of training samples / batch size. Which means that the training process will iterate a number of steps that is equal to one-fourth of the total number of samples in the training dataset during each epoch. The training will be executed for 5 epochs as we set epochs = 5. We also use callbacks. Which will monitor the validation accuracy and save the model which is responsible for maximum validation accuracy. We also store the training history in a variable and visualize it on a graph Figure 4.12. The figure represents two graphs one is training_accuracy Vs validation_accuracy and the other one is training_loss Vs validation_loss.

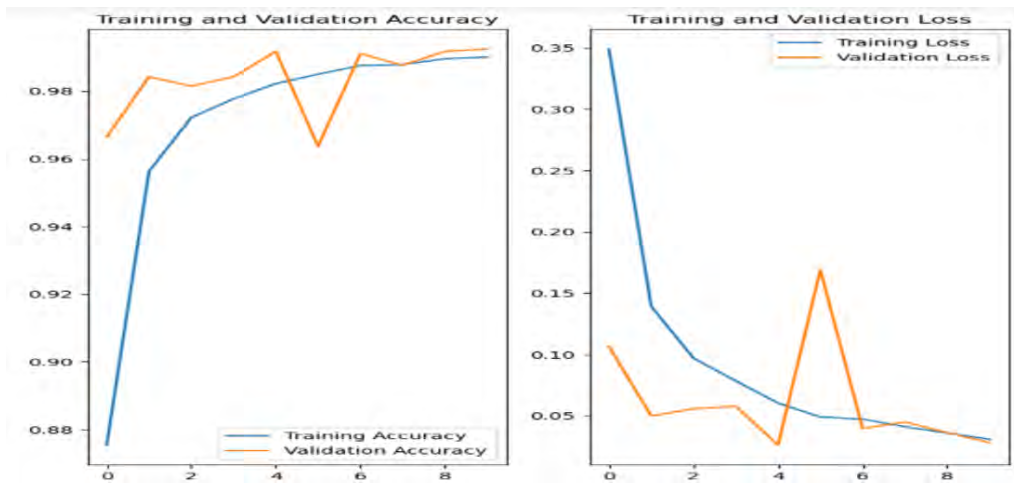


Figure 4.12: Training_accuracy Vs Validation_accuracy & Training_loss Vs Validation_loss

Then we evaluate the model with the test dataset. The performance metrics of evaluation are given below Table 4.11.

Table 4.11: Summary of performance matrices of evaluation

Metric	Value
Accuracy	0.8970
Loss	1.2697

Chapter 5

Result Analysis

5.1 Comparison between our design model (concentrateNet1,2,3) and the pre-trained model

Using precision, accuracy, recall, and f1-score, we evaluate ConcentrateNet1, concentrateNet2, ConcentrateNet3, and the pre-trained model named VGG16, InceptionV3, ResNet50, MobileNetV2. The results are shown in Table 5.1 Table 5.2

Table 5.1: Shows the comparison between our design models

Mod-els	Ac-cu-racy	Precision			Recall			F1-score			Specificity		
		At-ten-tive	Inat-ten-tive	Sle-epy	At-ten-tive	Inat-ten-tive	Sle-epy	At-ten-tive	Inat-ten-tive	Sle-epy	At-ten-tive	Inat-ten-tive	Sle-epy
Con-cen-trate Net1	93.1	95.2	90.1	94.1	90.3	95.9	93.2	92.7	92.9	93.7	97.6	95	97.1
Con-cen-trate Net2	92.7	94.6	91.6	91.9	90.7	92.9	94.6	92.6	92.3	93.2	97.3	95.9	95.8
Con-cen-trate Net3	92.2	96.3	91.5	89.2	88.8	91.7	96.1	92.4	91.6	92.5	98.2	95.9	94.1

Table 5.2: Shows the comparison between pre-trained models

Mod-els	Ac-cu-racy	Precision			Recall			F1-score			Specificity		
		At-ten-tive	Inat-ten-tive	Sle-epy	At-ten-tive	Inat-ten-tive	Sle-epy	At-ten-tive	Inat-ten-tive	Sle-epy	At-ten-tive	Inat-ten-tive	Sle-epy
VGG 16	89.5	92.7	86.3	89.7	86.6	89.1	92.8	89.6	87.7	91.2	96.4	93.3	94.6
Mo-bile NetV2	85.8	78.6	96.2	87.9	97.7	66.5	92.2	87.1	78.6	90.0	86.1	98.8	93.6
Res Net 50	90.4	94.8	93.2	84.7	86.8	86.3	98.1	90.7	89.6	90.9	97.5	97.0	91.1
In-cep-tion V3	89.7	89.9	90.7	88.8	91.4	81.4	95.9	90.7	85.7	92.2	94.7	95.9	93.9

5.2 Class-wise study for the Confusion matrix

Confusion matrix predicts the measures of the performance of a machine learning model on test data. It shows the number of accurate and inaccurate instances from the model's prediction. It is commonly used in classification models. The confusion matrix can be broken down into four outcomes:

- TP (True Positive): The model correctly predicted as positive instances as positive. For example, it is predicting a Dog when it is a Dog.
- TN (True Negative): The model correctly predicted negative instances as negative. For example, it is predicting Not Dog when it is Not Dog
- FP (False Positive): The model incorrectly predicted negative instances as positive. Suppose, the model is predicting a Dog while it is a Not Dog.
- FN (False Negative): The model incorrectly predicted positive instances as negative. Suppose, the model is predicting a Not Dog while it is a Dog.

For our research scenario, we categorize the confusion matrix outcomes into two: True and False Values. let's delve into understanding the confusion matrices below:

ConcentrateNet1:

		Predicted Classes		
		Attentive	Inattentive	Sleepy
Actual Values	Attentive	474	28	23
	Inattentive	13	475	7
	Sleepy	11	24	480

Figure 5.1: ConcentrateNet1 confusion matrix

Let's explain the first value of the matrix. In the vertical direction, we set 'Actual Class' and in the horizontal direction, we set 'Predicted Class'. From the matrix demonstrates (Figure 5.1) that in the attentive row, the total actual attentive frames are $474+28+23 = 525$. In the first column, the model predicted 474 attentive frames among 525 attentive frames, 13 inattentive frames as attentive frames, and 11 sleepy frames as attentive frames.

The second row illustrates that the total actual inattentive frames are $13+475+7 = 495$. In the second column, the model predicted 28 attentive frames as inattentive frames, 475 inattentive frames as inattentive frames, and 24 sleepy frames as inattentive frames.

The third row depicts that the total actual sleepy frames are $11+24+480 = 415$. In the third column, the model predicted 23 attentive frame as a sleepy frame, 7 inattentive frame as a sleepy frame, and 480 sleepy frames as sleepy frames.

Now we can also summarize by saying that in this confusion matrix, all the diagonal values are true predictions and others are false predictions.

ConcentrateNet2:

Likewise, in this confusion matrix (Figure 5.2), all the diagonal values are True predictions and others are False predictions of the model.

- True predictions: 476,460,487(Diagonal light blue boxes)
- False predictions: 21,28,20,15,7,21 (row-wise-All the bottle green boxes)

		Predicted Classes		
		Attentive	Inattentive	Sleepy
Actual Values	Attentive	476	21	28
	Inattentive	20	460	15
	Sleepy	7	21	487

Figure 5.2: ConcentrateNet2 confusion matrix

ConcentrateNet3:

		Predicted Classes		
		Attentive	Inattentive	Sleepy
Actual Values	Attentive	466	28	31
	Inattentive	12	454	29
	Sleepy	6	14	495

Figure 5.3: ConcentrateNet3 confusion matrix

Similarly to the other confusion matrices, the diagonal values of this matrix (Figure 5.3) are the true predictions and others are False predictions of the model.

- True predictions: 466,454,495(Diagonal light blue boxes)
- False predictions: 28,31,12,29,6,14 (row wise-All the bottle green boxes)

VGG16:

		Predicted Classes		
		Attentive	Inattentive	Sleepy
Actual Values	Attentive	455	47	23
	Inattentive	22	441	32
	Sleepy	14	23	478

Figure 5.4: VGG16 confusion matrix

In (Figure 5.4) matrix, all the diagonal values of this matrix are the true predictions and others are the False predictions of the model.

- True predictions: 455,441,478(Diagonal light blue boxes)
- False predictions: 47,23,22,32,14,23 (All the bottle green boxes)

InceptionV3:

		Predicted Classes		
		Attentive	Inattentive	Sleepy
Actual Values	Attentive	480	27	18
	Inattentive	48	403	44
	Sleepy	6	15	494

Figure 5.5: InceptionV3 confusion matrix

Here again, all the diagonal values of this matrix (Figure 5.5) are the true predictions and others are False predictions of the model.

- True predictions: 480,403,494(Diagonal light blue boxes)
- False predictions: 27,18,48,44,6,15 (All the bottle green boxes)

ResNet50:

		Predicted Classes		
		Attentive	Inattentive	Sleepy
Actual Values	Attentive	456	26	43
	Inattentive	20	427	48
	Sleepy	5	5	505

Figure 5.6: ResNet50 confusion matrix

In the same way, all the diagonal values of this matrix (Figure 5.6) are the true predictions and others are False predictions of the model.

- True predictions: 456,427,505(Diagonal light blue boxes)
- False predictions: 26,43,20,48,5,5 (All the bottle green boxes)

MobileNetV2:

		Predicted Classes		
		Attentive	Inattentive	Sleepy
Actual Values	Attentive	513	1	11
	Inattentive	112	329	54
	Sleepy	28	12	475

Figure 5.7: MobileNetV2 confusion matrix

In the same way, all the diagonal values of this matrix are the true predictions and others are False predictions of the model.

- True predictions: 513, 329, 475(Diagonal light blue boxes)
- False predictions: 1,11,112,54,28,12 (All the bottle green boxes)

5.3 Output

5.3.1 Analysing our Design model (Concentrate1, Concentrate2, Concentrate3) and the pre-trained model

Pre-trained models:

In this research paper, we used VGG16, InceptionV3, ResNet50, and MobileNetV2 as pre-trained models. These models have low accuracy comparing with our models. Moreover, the Precision, Recall, F1 score, and Specificity values of these models are decent. However, the computational costs are also high. This is because of the huge number of neurons and dense layers. The formulas we have used to find these values are given below:

1. $\text{accuracy} = \text{TP} / (\text{TP} + \text{FP} + \text{FN} + \text{TN})$
2. $\text{precision} = \text{T} / (\text{TP} + \text{FP})$
3. $\text{recall} = \text{TP} / (\text{TP} + \text{FN})$
4. $\text{f1_score} = 2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$
5. $\text{specificity} = \text{TN} / (\text{TN} + \text{FP})$

Here, TP = True positive, TN = True negative, FN = False negative, FP = False positive.

5.3.2 Our Design models:

We used 3 designed models for our research purpose. These models are very lightweight and the computational costs are also very low. Because it has fewer neurons and less number of dense layers. Our three models: ConcentrateNet1, ConcentrateNet2, and ConcentrateNet3 have higher accuracy than pre-trained models. It indicates that our models' performance is very good. So these models are very preferable.

Table 5.3: shows the summary of the ConcentrateNet1 model

<i>Layer</i>	<i>Output Shape</i>	<i>Parameters</i>
conv2d (Conv2D)	(None,100,100,16)	448
max_pooling2d (MaxPooling2D)	(None,50,50,16)	0
conv2d_1 (Conv2D)	(None,50,50,32)	4640
max_pooling2d_1 (MaxPooling2D)	(None,25,25,32)	0
conv2d_2 (Conv2D)	(None,25,25,64)	18496
max_pooling2d_2 (MaxPooling2D)	(None,12,12,64)	0
flatten (Flatten)	(None,9216)	0
dense (Dense)	(None,512)	4719104
dropout (Dropout)	(None,512)	0
dense_1 (Dense)	(None,3)	1539
Total parameters		4744227
Trainable Parameters		4744227
Non-trainable parameters		0

Table 5.4: Shows the summary of the ConcentrateNet2 model

<i>Layer</i>	<i>Output Shape</i>	<i>Parameters</i>
conv2d (Conv2D)	(None,98,98,16)	448
max_pooling2d (MaxPooling2D)	(None,49,49,16)	0
batch_normalization (BatchNormalization)	(None,49,49,16)	64
conv2d_1 (Conv2D)	(None,47,47,32)	4640
max_pooling2d_1 (MaxPooling2D)	(None,23,23,32)	0
batch_normalization_1 (BatchNormalization)	(None,23,23,32)	128
conv2d_2 (Conv2D)	(None,21,21,64)	18496
max_pooling2d_2 (MaxPooling2D)	(None,10,10,64)	0
batch_normalization_2 (BatchNormalization)	(None,10,10,64)	256
flatten (Flatten)	(None,6400)	0
dense (Dense)	(None,128)	819328
dropout (Dropout)	(None,128)	0
dense_1 (Dense)	(None,3)	387
Total parameters		843747
Trainable Parameters		843523
Non-trainable parameters		224

Table 5.5: Shows the summary of the ConcentrateNet3 model

<i>Layer</i>	<i>Output Shape</i>	<i>Parameters</i>
conv2d (Conv2D)	(None,100,100,16)	448
batch_normalization (Batch-Normalization)	(None,100,100,16)	64
average_pooling2d (Average-Pooling2D)	(None,50,50,16)	0
conv2d_1 (Conv2D)	(None,50,50,32)	4640
batch_normalization_1 (Batch Normalization)	(None,50,50,32)	128
average_pooling2d (Average-Pooling2D)	(None,25,25,32)	0
conv2d_2 (Conv2D)	(None,25,25,64)	18496
batch_normalization_2 (BatchNormalization)	(None,25,25,64)	256
average_pooling2d (Average-Pooling2D)	(None,12,12,64)	0
flatten (Flatten)	(None,9216)	0
dense (Dense)	(None,512)	4719104
dropout (Dropout)	(None,512)	0
dense_1 (Dense)	(None,3)	1539
Total parameters		4719104
Trainable Parameters		4719104
Non-trainable parameters		224

5.3.3 Visual Representation and Analysis of the Result Implementation

We divided our dataset into three categories. They are Attentive, Inattentive and Sleepy. We used 10% of our total dataset as a test dataset. In the test dataset, the count of attentive frames = 525, inattentive frames = 495, and sleepy = 515. Percentage values are 34.20%, 32.24%, and 33.55% respectively. Analyzing each model against these values through the bar chart and pie chart provided below.

ConcentrateNet1:

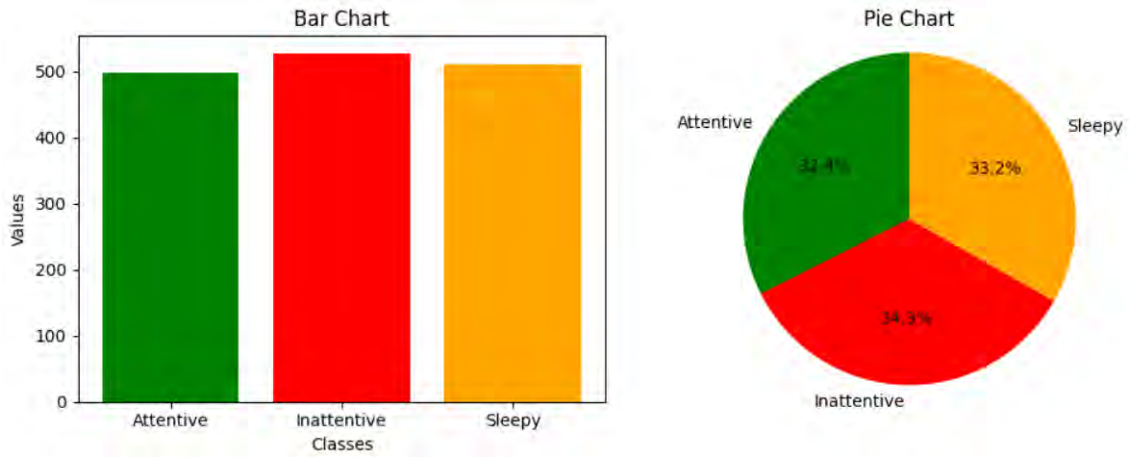


Figure 5.8: Graphical representation of ConcentrateNet1 model result

Table 5.6: summary of the ConcentrateNet1 model result

	Actual Test Dataset Values			ConcentrateNet1 predicted value		
	Atten- tive	inatten- tive	Sleepy	Atten- tive	inatten- tive	Sleepy
Frames in number	525	495	515	498	527	510
Percentage	34.20%	32.25%	33.55%	32.44%	34.33%	33.22%

In elaboration, the actual attentive test dataset value was 525. However, the model predicted 498 frames as attentive. It accurately predicted 474 frames as attentive (as depicted in Figure 5.1) and predicted the rest of the 24 frames incorrectly. So we can easily say that our model ConcentrateNet1's prediction and performance are very good and accurate.

ConcentrateNet2:

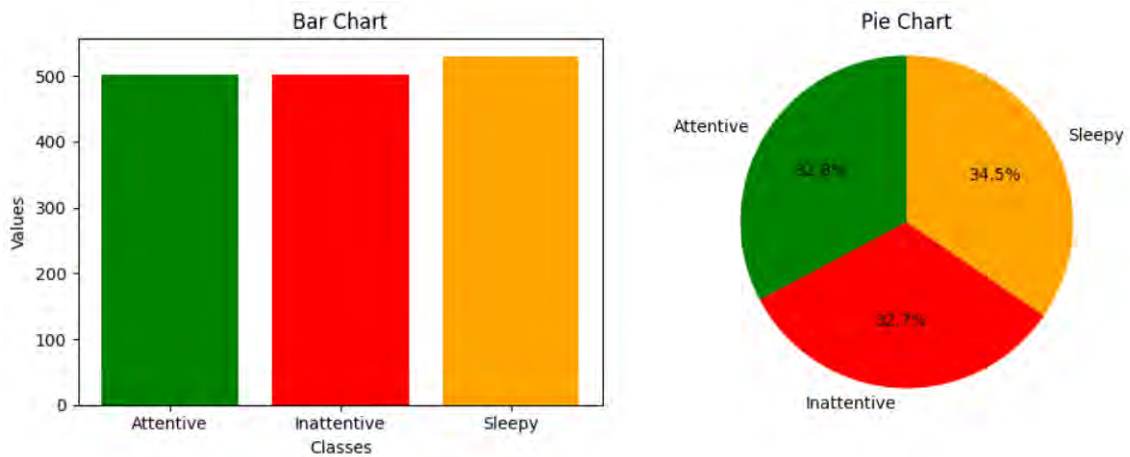


Figure 5.9: Graphical representation of ConcentrateNet2 model result

Table 5.7: Shows the summary of the ConcentrateNet2 model result

	Actual Test Dataset Values			ConcentrateNet2 predicted value		
	Atten- tive	inatten- tive	Sleepy	Atten- tive	inatten- tive	Sleepy
Frames in number	525	495	515	503	502	530
Percentage	34.20%	32.25%	33.55%	32.77%	32.70%	34.53%

In elaboration, the actual attentive test dataset value was 525. However, the model predicted 503 frames as attentive. It accurately predicted 476 frames as attentive (as depicted in Figure 5.2) and predicted one frame incorrectly. So we can easily say that 27 model ConcentrateNet2's prediction and performance are very good and accurate.

ConcentrateNet3:

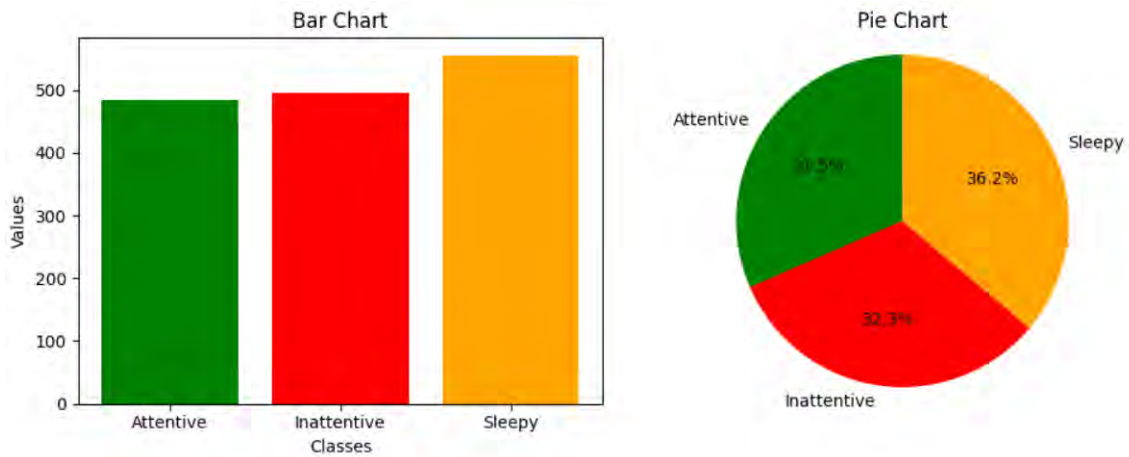


Figure 5.10: Graphical representation of ConcentrateNet3 model result

Table 5.8: Shows the summary of the ConcentrateNet3 model result

	Actual Test Dataset Values			ConcentrateNet2 predicted value		
	Atten- tive	inatten- tive	Sleepy	Atten- tive	inatten- tive	Sleepy
Frames in number	525	495	515	484	496	555
Percentage	34.20%	32.25%	33.55%	31.53%	32.31%	36.16%

In elaboration, the actual attentive test dataset value was 525. However, the model predicted 484 frames as attentive. It accurately predicted 466 frames as attentive (as depicted in Figure 5.3) and predicted the rest of the 18 frames incorrectly. So we can easily say that our model ConcentrateNet3's prediction and performance are very good and accurate.

VGG16:

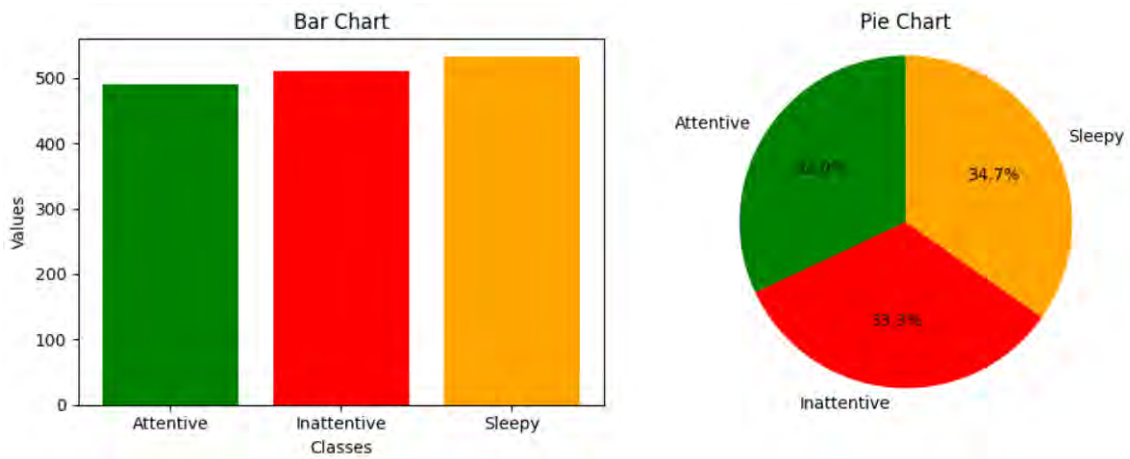


Figure 5.11: Graphical representation of VGG16 model result

Table 5.9: Shows the summary of the VGG16 model result

	Actual Test Dataset Values			VGG16 predicted value		
	Atten- tive	inatten- tive	Sleepy	Atten- tive	inatten- tive	Sleepy
Frames in number	525	495	515	491	511	533
Percentage	34.20%	32.25%	33.55%	31.99%	33.29%	34.72%

In elaboration, the actual attentive test dataset value was 525. However, the model predicted 491 frames as attentive. It accurately predicted 455 frames as attentive (as depicted in Figure 5.4) and predicted the rest of the 36 frames incorrectly. So we can easily say that our model VGG16's prediction and performance are very good and accurate.

InceptionV3:

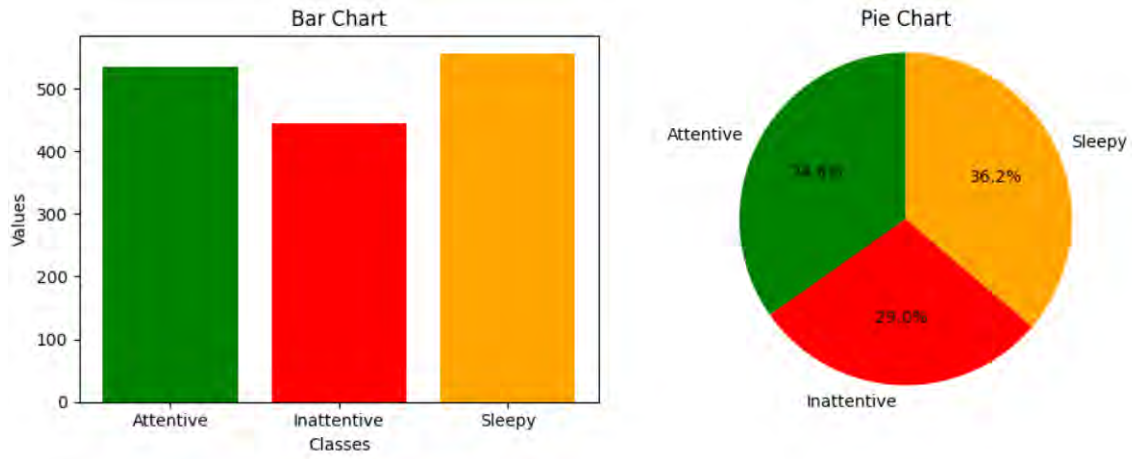


Figure 5.12: Graphical representation of InceptionV3 model result

Table 5.10: Shows the summary of the InceptionV3 model result

	Actual Test Dataset Values			InceptionV3 predicted value		
	Atten- tive	inatten- tive	Sleepy	Atten- tive	inatten- tive	Sleepy
Frames in number	525	495	515	534	445	556
Percentage	34.20%	32.25%	34.79%	31.27%	28.99%	36.22

In elaboration, the actual attentive test dataset value was 525. However, the model predicted 534 frames as attentive. It accurately predicted 480 frames as attentive (as depicted in Figure 5.5) and predicted 54 frames incorrectly. So we can easily say that our model InceptionV3's prediction and performance are very good and accurate.

ResNet50:

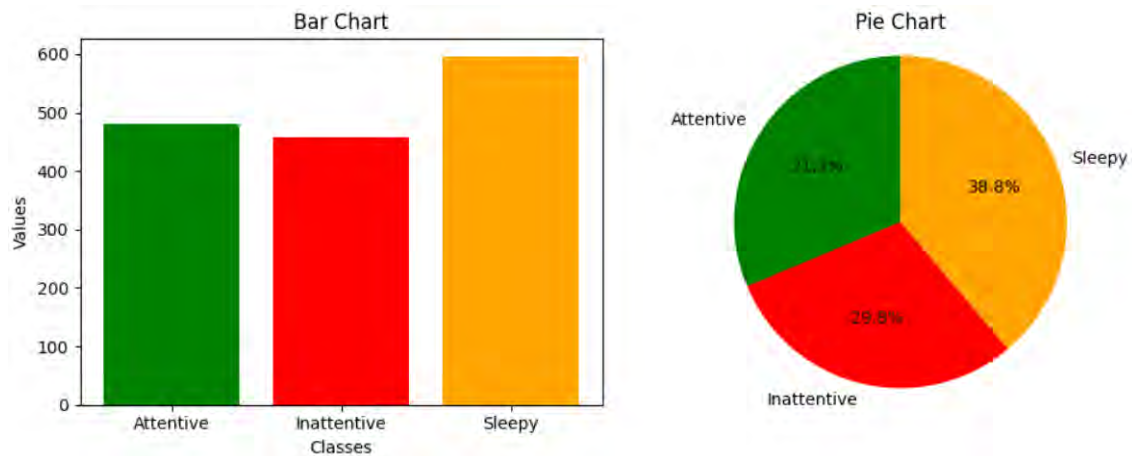


Figure 5.13: Graphical representation of ResNet50 model result

Table 5.11: Shows the summary of the ResNet50 model result

	Actual Test Dataset Values			RetNet50 predicted value		
	Atten- tive	inatten- tive	Sleepy	Atten- tive	inatten- tive	Sleepy
Frames in number	525	495	515	481	458	596
Percentage	34.20%	32.25%	33.55%	31.34%	29.84%	38.83%

In elaboration, the actual attentive test dataset value was 525. However, the model predicted 481 frames as attentive. It accurately predicted 456 frames as attentive (as depicted in Figure 5.6) and predicted 25 frames incorrectly. So we can easily say that our model RetNet50's prediction and performance are very good and accurate.

MobileNetV2:

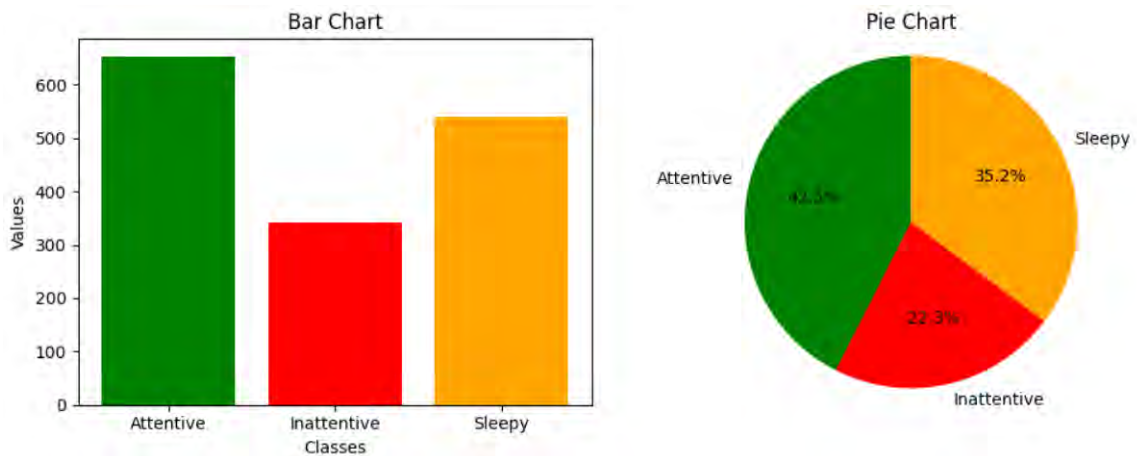


Figure 5.14: Graphical representation of MobileNetV2 model result

Table 5.12: Shows the summary of the MobileNetV2 model result

	Actual Test Dataset Values			MobileNetV2 predicted value		
	Atten- tive	inatten- tive	Sleepy	Atten- tive	inatten- tive	Sleepy
Frames in number	525	495	515	653	342	540
Percentage	34.20%	32.25%	33.55%	42.54%	22.28%	35.18%

In elaboration, the actual attentive test dataset value was 525. However, the model predicted 653 frames as attentive. It accurately predicted 513 frames as attentive (as depicted in Figure 5.7) and predicted the rest of the 140 frames incorrectly. So we can easily say that our model MobileNetV2's prediction and performance are very good and accurate.

Chapter 6

Conclusion

This paper is focused on measuring the concentration levels of the students at online platforms for a better understanding of their course outcome. To do so we used our design model (concentrateNet1, ConcentrateNet2, ConcentrateNet3) and pre-trained model (VGG16, InceptionV3, ResNet50, MobileNetV2). We selected these models based on their lightweight and low computational cost. During the process of data pre-processing we categorized our classes into three: “Attentive”, “Inattentive”, and “Sleepy”. By evaluating these things we provide the two work areas of this research which are how good the courses of online platforms are and whether the concentration of the students’ affects their results and dropouts.

6.1 Future work

The research work we are proposing is to measure students’ concentration levels to analyze their poor outcomes in online courses. As nobody worked on this area based on students’ concentration level we had to create our dataset and progress the process. Subsequently, it also creates new areas for the researchers to work further. Future works that we can work on:

- Collect students’ reviews on course materials and compare them with their concentration to perfectly analyze the problem.
- We can update the dataset with more variants. Also, we will use videos instead of frames.
- We can add more sub-categorized classes.

Bibliography

- [1] G. Goudelis, A. Tefas, and I. Pitas, “Automated facial pose extraction from video sequences based on mutual information,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 3, pp. 418–424, 2008.
- [2] S.-H. Zhong, Y. Li, Y. Liu, and Z. Wang, “A computational investigation of learning behaviors in moocs,” *Computer Applications in Engineering Education*, vol. 25, no. 5, pp. 693–705, 2017.
- [3] J. Hernandez, F. Rodriguez, I. Hilliger, and M. Perez-Sanagustin, “Moocs as a remedial complement: Students’ adoption and learning outcomes,” *IEEE Transactions on Learning Technologies*, vol. 12, no. 1, pp. 133–141, 2018.
- [4] J. Hernandez, F. Rodriguez, I. Hilliger, and M. Perez-Sanagustin, “Moocs as a remedial complement: Students’ adoption and learning outcomes,” *IEEE Transactions on Learning Technologies*, vol. 12, no. 1, pp. 133–141, 2018.
- [5] J. Liao, J. Tang, and X. Zhao, “Course drop-out prediction on mooc platform via clustering and tensor completion,” *Tsinghua Science and Technology*, vol. 24, no. 4, pp. 412–422, 2019.
- [6] Z. Xie, “Assessing the attractions of moocs from the perspective of scientometrics,” *IEEE Access*, vol. 7, pp. 136 409–136 418, 2019.
- [7] Z. Xie, “Modelling the dropout patterns of mooc learners,” *Tsinghua Science and Technology*, vol. 25, no. 3, pp. 313–324, 2019.
- [8] B. Wu and P. Li, “Influence of moocs ewom on the number of registrations and completions,” *IEEE Access*, vol. 8, pp. 158 826–158 838, 2020.
- [9] S. Yin, L. Lei, H. Wang, and W. Chen, “Power of attention in mooc dropout prediction,” *IEEE Access*, vol. 8, pp. 202 993–203 002, 2020.
- [10] Y. Zheng, Z. Gao, Y. Wang, and Q. Fu, “Mooc dropout prediction using fwts-cnn model based on fused feature weighting and time series,” *IEEE Access*, vol. 8, pp. 225 324–225 335, 2020.
- [11] D. Cao, J. Liu, L. Hao, W. Zeng, C. Wang, and W. Yang, “Recognition of students’ behavior states in classroom based on improved mobilenetv2 algorithm,” *The International Journal of Electrical Engineering Education*, vol. 60, p. 002 072 092 199 659, Mar. 2021. DOI: 10.1177/0020720921996595.
- [12] G. Kostopoulos, T. Panagiotakopoulos, S. Kotsiantis, C. Pierrakeas, and A. Kameas, “Interpretable models for early prediction of certification in moocs: A case study on a mooc for smart city professionals,” *IEEE Access*, vol. 9, pp. 165 881–165 891, 2021.

- [13] C. Qi and S. Liu, “Evaluating on-line courses via reviews mining,” *IEEE Access*, vol. 9, pp. 35 439–35 451, 2021.
- [14] J. Zhang, M. Gao, and J. Zhang, “The learning behaviours of dropouts in moocs: A collective attention network perspective,” *Computers & education*, vol. 167, p. 104 189, 2021.
- [15] K. Sumino, I. Mitsugami, and R. Sagawa, “Expression-controllable facial video generation for impression quantification,” in *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE)*, IEEE, 2022, pp. 469–470.
- [16] A. Arwa, K. Altuwairqi, K. Salma, A. Nihal, and S. Alkhurairji, “Cnn-based face emotion detection and mouse movement analysis to detect student’s engagement level,” in Jun. 2023, pp. 604–626, ISBN: 978-3-031-26383-5. DOI: 10.1007/978-3-031-26384-2_53.
- [17] M. Chen, Y.-C. Chen, T.-Y. Chou, and F.-S. Ning, “Pm2.5 concentration prediction model: A cnn-rf ensemble framework,” *International Journal of Environmental Research and Public Health*, vol. 20, Feb. 2023. DOI: 10.3390/ijerph20054077.
- [18] T. Hidayat, I. Astuti, A. Yaqin, A. Tjilen, and T. Arifianto, “Grouping of image patterns using inceptionv3 for face shape classification,” *JOIV : International Journal on Informatics Visualization*, vol. 7, Dec. 2023. DOI: 10.30630/joiv.7.4.1743.
- [19] D. Hindarto, N. Afarini, and E. H, “Comparison efficacy of vgg16 and vgg19 insect classification models,” *JIKO (Jurnal Informatika dan Komputer)*, vol. 6, pp. 189–195, Dec. 2023. DOI: 10.33387/jiko.v6i3.7008.
- [20] L. Sha, E. Fincham, L. Yan, *et al.*, “The road not taken: Preempting dropout in moocs,” in *International Conference on Artificial Intelligence in Education*, Springer, 2023, pp. 164–175.
- [21] M. Shahabi, B. Nobakhsh, A. Shalhaf, R. Rostami, and R. Kazemi, “Prediction of treatment outcome for repetitive transcranial magnetic stimulation in major depressive disorder using connectivity measures and ensemble of pre-trained deep learning models,” *Biomedical Signal Processing and Control*, vol. 85, Mar. 2023. DOI: 10.1016/j.bspc.2023.104822.