Faculty of Engineering
Biomedical Engineering Department
Graduation project 2023

A wearable Behavioral Aid for Autistic Children

Supervisor:

- DR. Ibrahim Sadek

Prepared by:

- Abdulrhman Nasser
- Merna Ahmed Mansour
- Shaimaa Abd Elkhalek

Acknowledgment

In the name of Allah, the most merciful, all praise him for his strength &blessings in completing this project.

Firstly, we would like to express our sincere appreciation and gratitude to oursupervisor, Professor Ibrahim, for his enthusiastic support, unwavering patience, and continuous provision of helpful information that have been instrumental in the successful completion of this project.

Secondly, we would like to express our sincere appreciation and gratitude tothe Information Technology Industry Development Agency (ITIDA) for supporting our graduation project through their Graduation Projects Support program.

Their financial support has been instrumental in enabling us to complete this project successfully. We are grateful for the opportunity to have received this support, which has significantly contributed to our academic and personal growth.

Finally, we would like to pass our deepest regards to our families & friends who helped & pushed us to our limits all these college years &successfullyfinished this graduation project. Words are not enough to state our deepest thanks & gratitude for them.
May Allah grant us the needed success in return for their efforts & sacrificesfor all these years.

Abstract

ASD is a developmental disorder that significantly impacts communication, social interaction, and behavior. The prevalence of ASD has been increasing in recent years, emphasizing the need for effective support and interventions. This project aims to address the challenges faced by individuals with Autism Spectrum Disorder (ASD) in social communication and interaction.

To meet this objective, we propose the design and development of a wearable Behavioral Aid for Autistic Children, which combines glasses integrated with a mobile application and an emotion recognition predictive model. The glasses will incorporate a camera to capture facial expressions of individuals around the person with ASD, while the mobile application will utilize deep learning algorithms for real-time analysis. A predictive model will be trained to recognize a range of emotions and provide instant feedback through the glasses, thereby enhancing the social communication and interaction skills of individuals with ASD.

To achieve accurate emotion recognition, a baseline convolutional neural network (CNN) model serves as the foundation. Considering the benefits of transfer learning in improving model performance, we investigate the effectiveness of reducing the complexity of the baseline model to achieve comparable accuracy with a less advanced model that utilizes transfer learning. This approach aims to optimize model performance while minimizing resource usage.

Through the implementation of transfer learning, we observe significant improvements in the model's performance. Among the evaluated models, the MobileNetV2 model demonstrates the best performance in real-world scenarios, while the EfficientNetB7 model achieves the highest overall accuracy. The model is connected to a simple mobile application via Bluetooth to help autistic children with facial expression recognition. It also could include educational and other features aimed at improving children's communication skills.

The outcomes of this project highlight the potential of wearable technology and deep learning in supporting individuals with ASD. The developed Behavioral Aid for Autistic Children provides real-time social cues, facilitating better understanding of emotions and improved social interaction skills. This project contributes to ongoing research in supporting individuals with ASD and lays the foundation for future advancements in wearable technologies and emotion recognition systems.

Keywords: Autism Spectrum Disorder, deep learning, emotion recognition, transfer learning, wearable technology, social communication

Table of Contents

Acknowledgment 2
Abstract 3
Table of Contents 4
List of figures 7
List of Tables.....9
Chapter 1 10
Introduction 10
1.1 Problem definition.....11
1.2 Project Objective.....12
1.3 Book index 13
Chapter 2 14
Theoretical background.....14
2.1 Autism spectrum disorder.....15
2.1.1 Signs and Symptoms of ASD 15
2.1.2 People on the autism spectrum also may have many strengths,including:.....17
2.1.3 Causes and related factors.....17
2.1.4 Diagnosing ASD 18
2.1.5 Treatments and therapies 20
2.1.6 Medication 20
2.1.7 Behavioral, psychological, and educational interventions 20
2.1.8 ASD statistics.....21
2.2 Emotional categories.....22
2.3 Data processing 24
2.4 Artificial intelligence 25
2.4.1 Deep Learning.....26
2.4.2 Selecting ML dataset. 27
2.4.3 Convolutional Neural Network.....28

2.4.3.1 CNN Parameters……..……...30
2.4.4 Transfer learning.....29
2.5 Mobile application 30
2.5.1 Different Types of Mobile Apps.....31
2.5.2 Frontend App Development 32
2.5.3 Backend Development.....32
2.5.4 Android operating system.....33
Chapter 3 34
Literature Review.....34
Chapter 4 38
Methodology 38
4.1 Block diagram.....39
4.2 Dataset.....40
4.2.1 CK+ Dataset.....40
4.2.2 AffectNet Dataset.....41
4.2.3 Dataset description (FER2013) *chosen dataset*.....42
4.3 Pre-processing.....44
4.3.1 Balancing techniques 44
4.3.2 Data augmentation 48
4.3.3 Preparing data for model after balancing.....51
4.4. Building Sequential CNN model 52
4.5 Building transfer learning models.....57
4.6 Extracting results.....63
4.7 Predicting on new dataset. 67
4.8 Saving weights of used model. 68
4.9 Real time prediction.....68
4.10 mobile application.....69
4.10.1 Developing Main activity. 69
4.10.2 Android UI Layouts 71
4.10.3 Android UI component 71
4.10.4 Developing Fragment 74
4.10.5 Bottom Navigation Bar Android: 76

4.10.6 Settings fragment 77
4.10.7 Home fragment 78
4.10.8 Education fragment 79
4.11 Hardware connection 80
4.11.1 Camera module 80
4.11.2 Arduino uno 80
4.11.3 Powering of Esp 32 camera 81
4.11.4 Total Cost 81
4.11.5 Connection between camera and model 81
4.11.6 Connection between model and application 82
Chapter 5 84
Results & Discussion 84
5.1 Dataset 85
5.2 Pre-processing phase 85
5.3 Model phase 86
5.3.1 CNN Model 86
5.3.2 Pretrained models 91
5.4. Predictions on unseen datasets 103
5.5 mobile application 107
5.5.1 Main activity 107
5.5.2 Home fragment 108
5.5.3 Setting fragment 108
5.5.4 Education fragment 109
5.5.5 Problems and Solutions 110
Chapter 6 111
Conclusion & Future work 111
6.1 Conclusion 112
6.2 Future work 112
References 113

List of figures

Figure 1: Signs of autism 17
Figure 2: ASD statistics 21
Figure 3: Ekman's 6 Basic Emotions. From Ekman's six Basic Emotions and How They Affect Our Behavior, by Harrison 2021.....23
Figure 4: Action Units of facial expressions. From Facial expressions, by Lansley 2022.. 23
Figure 5: Correlation between AI, ML and DL. From Is machine learning required for deep learning? by Suman 2019. 26
Figure 6: Deep learning vs Machine learning.....26
Figure 7: Block diagram of overall system.....39
Figure 8: CK+ Dataset 40
Figure 9: AffectNet Dataset 42
Figure 10: FER2013 dataset.....43
Figure 11: distribution for each of the seven emotions followed by a percentage. 43
Figure 12: distribution for FER2013 dataset before balancing.....44
Figure 13: GENERATOR.....45
Figure 14: DISCRIMINATOR 46
Figure 15: GANs the first trial 46
Figure 16: GANs the second trial 47
Figure 17: upsampling.....48
Figure 18: Data augmentation techniques.....49
Figure 19: augmentation techniques 50
Figure 20: distribution of dataset after balancing 51
Figure 21: Image to array conversion 52
Figure 22 : Illustration of our 4-layer built-from-scratch DNN model. Each layer is composed of convolutions, max-pooling, and a Rectified Linear Unit (ReLU) activation function. The parameterization of the convolutional layer is denoted as" Channels @ width _ height".....54
Figure 23: Illustration of the CNN Sequential model.....56
Figure 24: The pre-trained VGG16 model with weights 'imagenet' illustration of each layer. The trainable layers are freezed, and one Flatten, Dense (512) and Dense(7) layers are added at the end.....58
Figure 25: ResNet50 model architecture 59
Figure 26: DenseNet201 model architecture 60
Figure 27: MobileNet model architecture.....61
Figure 28 The EfficientNet family compared to other ImageNet models. 62
Figure 29: EfficientNetB0 model architecture.....63

Figure 30: EfficientNetB7 model architecture.....63
Figure 31: Confusion matrix.....65
Figure 32: High loss in the left model; low loss in the right model. 66
Figure 33: **Activity lifecycle**.....70
Figure 34: Android UI component.....73
Figure 35: Fragment.....74
Figure 36: **lifecycle of Fragment**.....75
Figure 37: Camera module.....80
Figure 38: Arduino uno.....80
Figure 39: powering components.....81
Figure 40: the connection of Arduino with esp32 camera……..……82
Figure 41: Esp32 camera and mobile application communication…..……83
Figure 42: glasses.....83
Figure 43: accuracy and loss history for CNN model (Before applying processing).....87
Figure 44: confusion matrix for CNN model during 80 epochs before preprocessing. ...88
Figure 45: accuracy and loss history for CNN model (After applying processing) 89
Figure 46: confusion matrix for CNN model during 50 epoch after preprocessing……90
Figure 47: accuracy and loss history for VGG16 model 91
Figure 48: confusion matrix for VGG16 pre_trained model during 50 epochs 92
Figure 49: accuracy and loss history for ResNet50 model 93
Figure 50: confusion matrix for ResNet50 model during 50 epoch 94
Figure 51: accuracy and loss history for DenseNet201 model 95
Figure 52: confusion matrix for DenseNet201 model during 50 epoch 96
Figure 53: accuracy and loss history for MobileNetV2 model.....97
Figure 54: confusion matrix for MobileNetV2 model during 50 epochs 98
Figure 55: accuracy and loss history for EfficientNetB0 model 99
Figure 56: confusion matrix for EfficientNetB0 model during 50 epochs.....100
Figure 57: accuracy and loss history for EfficientNetB7 model 101
Figure 58: confusion matrix for EfficientNetB7 model during 50 epoch 102
Figure 59: Predictions on unseen datasets 104
Figure 60: model predictions using laptop……...105
Figure 61: accuracy, precision, recall and f-score chart…….....106-107
Figure 62: mobile application logo 107
Figure 63: Home fragment.....108
Figure 64: Setting fragment. 109
Figure 65: Education fragment.....109

List of Tables

Table 1: showing Emotion Action Units.....23
Table 2: Pre trained models.....30
Table 3:Results of FACIAL EMOTION RECOGNITION USING DEEP LEARNING 35
Table 4: KDEF dataset performance. From ResNet-50 and VGG-16 for recognizing Facial Emotions by Dhankhar 2019.....36
Table 5: KDEF dataset performance with Transfer Learning. From ResNet-50 and VGG-16 for recognizing Facial Emotions by Dhankhar 2019. 36
Table 6: PREVIOUS WORK CLASSIFICATION AND ACCURACY 37
Table 7 list of input shape expected for each model.....62
Table 8: Total Cost.....88
Table 9: Classification report from using CNN model on 80 epochs before processing. 88
Table 10: Classification report from using CNN model on 50 epochs after processing. .90
Table 11: Classification report from using VGG16 model.....93
Table 12: Classification report from using ResNet50 model 95
Table 13: Classification report from using DenseNet201 model 97
Table 14: Classification report from using DenseNet201 model 99
Table 15: Classification report from using EfficientNetB0 model.....101
Table 16: Classification report from using EfficientNetB7 model.....103
Table 17: Predictions from testing 104
Table 18: problems and solutions.....110

Chapter 1

Introduction

Autism spectrum disorder (ASD) is a developmental disorder that affects communication, social interaction, and behavior. It is a growing concernin many countries, including the United States, where the prevalence of autism has increased dramatically in recent years. According to the Centers for Disease Control and Prevention (CDC), the most recent estimates suggest that about 1 in 36 children in the United States have ASD, which represents a significant increase from previous years.
This means that millions of families are affected by autism, and the number of individuals living with autism is likely to continue to rise. Inthis context, it is important to understand what can be done to support those who are living with ASD.

1.1 Problem definition

One of the core features of autism spectrum disorder (ASD) is difficulty with social communication and interaction. This can manifest as difficulty understanding social cues and nonverbal communication, such as facial expressions. They may struggle to interpret others' emotions and may have difficulty expressing emotions and making and maintaining social relationships.
This can lead to challenges in forming and maintaining friendships, difficulties in the workplace, and social isolation. These challenges can impact an individual's quality of life, leading to anxiety, depression, and other mental health issues. Therefore, it is crucial to provide individuals with ASD with interventions and therapies that can help them develop social communication and interaction skills, better understand emotions, and form meaningful relationships with others.

1.2 Project Objective

Our goal is to design and develop a wearable Behavioral Aid for Autistic Children consisting of a system of glasses that are integrated with a mobile application and a predictive model for emotion recognition. The glasses will be equipped with a camera that captures the facial expressions of those around the individual with Autism Spectrum Disorder (ASD), and the mobile application will use deep learning algorithms to analyze these expressions in real-time.

The predictive model will be trained to recognize a range of emotions and provide real-time feedback to the individual with ASD through the glasses. By providing real-time social cues, we aim to improve the social communication and interaction skills of individuals with ASD.

The system will be designed to be user-friendly, with an interface that is easy for both the individual with ASD and their caregivers to use. The system will be validated through collaboration with physicians and other experts in the field of autism research and treatment. There are several advantages to the system that we have proposed.

- Real-time social cue delivery: The system will provide real-time social cues to individuals with ASD, which can be critical in helping them navigate social situations and improve their social communication and interaction skills. By recognizing and interpreting the facial expressions of those around them, the system can provide feedback to the individual with ASD, helping them better understand social cues and respond appropriately.
- Increased independence: The system can help individuals with ASD become more independent in social situations, as they will have access to real-time feedback and social cues, they need to interact with others effectively. This can lead to increased confidence and self-esteem, which can have a positive impact on their overall well-being.

- Accessibility: The system is designed to be user-friendly and accessible to individuals with ASD, as well as their caregivers and family members. The glasses are lightweight and comfortable to wear, and the mobile application is designed to be easy to use. This ensures that the system can be used in a variety of settings, including home, school, and community settings.

Overall, the system has the potential to significantly improve the social communication and interaction skills of individuals with ASD, while also providing valuable data for research and personalized treatment.

1.3 Book index

In this book, there are 6 chapters:

- **Chapter 1**: Introduction.
- **Chapter 2:** Theoretical background.
- **Chapter 3**: Literature Review.
- **Chapter 4**: Methodology.
- **Chapter 5**: Results & Discussion.
- **Chapter 6:** Conclusion & future work.

Chapter 2

Theoretical background

2.1 Autism spectrum disorder

Autistic Spectrum Disorder is a severe, widespread developmental disorder for a person who has been accompanying the person throughout his or her life, affecting their perception, thinking and behavior. Although autism can be diagnosed at any age, it is described as a “developmental disorder” because symptoms generally appear in the first 2 years of life.
According to the Diagnostic and Statistical Manual of Mental Disorders (DSM-5), a guide created by the American Psychiatric Association that health care providers use to diagnose mental disorders, people with ASD often have:

- Significant difficulties in developing the person's social and communication skills and mutual interactions with those around him Restricted interests and repetitive behaviors.
- Symptoms that affect their ability to function in school, work, and other areas of life.

Autism is known as a “spectrum” disorder because there is wide variation in the type and severity of symptoms people experience.
People of all genders, races, ethnicities, and economic backgrounds can be diagnosed with ASD. Although ASD can be a lifelong disorder, treatments and services can improve a person’s symptoms and daily functioning. The American Academy of Pediatrics recommends that all children receive screening for autism. Caregivers should talk to their child’s health care provider about ASD screening or evaluation [1].
Autism Spectrum Disorder (ASD) is a lifelong developmental disorder that prevents people from understanding correctly what they see, hear, and generally feel. As a result, they face serious problems in their social relationships, communication, and behavior. Children with autism spectrum have significant difficulties in recognition, in understanding and expressing emotions. They tend to avoid human faces and is difficult to understand why facial features are "moving", changing, as a result, the inability to read the emotions in the human face weakens their ability to communicate with other people [1].

2.1.1 Signs and Symptoms of ASD

People with ASD may struggle with confined or repetitive activities or interests, as well as social communication and engagement.
Additionally, people with ASD could learn, move, or pay attention in various ways. It is crucial to remember that some individuals without ASD may also experience some of these symptoms. These traits can make life extremely difficult for those who have ASD

Social communication / interaction behaviors may include:

- Making little or inconsistent eye contact
- Appearing not to look at or listen to people who are talking.
- Infrequently sharing interest, emotion, or enjoyment of objects or activities (including infrequent pointing at or showing things to others)
- Not responding or being slow to respond to one's name or to other verbal bids for attention.
- Having difficulties with the back and forth of conversation
- Often talking at length about a favorite subject without noticing that others are not interested or without giving others a chance to respond.
- Displaying facial expressions, movements, and gestures that do not match what is being said.
- Having an unusual tone of voice that may sound sing-song or flat and robot-like
- Having trouble understanding another person's point of view or being unable to predict or understand other people's actions.
- Difficulties adjusting behaviors to social situations.
- Difficulties sharing in imaginative play or in making friends [1].

Restrictive / repetitive behaviors may include:

- Repeating certain behaviors or having unusual behaviors, such as repeating words or phrases (a behavior called echolalia)
- Having a lasting intense interest in specific topics, such as numbers, details, or facts
- Showing overly focused interests, such as with moving objects or parts of objects
- Becoming upset by slight changes in a routine and having difficulty with transitions.
- Being more sensitive or less sensitive than other people to sensory input, such as light, sound, clothing, or temperature
- People with ASD may also experience sleep problems and irritability [1].

Figure 1: Signs of autism

2.1.2 People on the autism spectrum also may have many strengths, including:

- Being able to learn things in detail and remember information for long periods of time.
- Being strong visual and auditory learners
- Excelling in math, science, music, or art [1].

2.1.3 Causes and related factors

Researchers don't know the primary causes of ASD, but studies suggest that a person's genes can act together with aspects of their environment to affect development in ways that lead to ASD. Some factors that are associated with an increased likelihood of developing ASD include:

- Having a sibling with ASD
- Having older parents
- Having certain genetic conditions (such as Down syndrome or Fragile X syndrome)
- Having a very low birth weight.

2.1.4 Diagnosing ASD

Health care providers diagnose ASD by evaluating a person's behavior and development. ASD can usually be reliably diagnosed by age 2. It is important to seek an evaluation as soon as possible. The earlier ASD is diagnosed, the sooner treatments and services can begin.

Diagnosis in young children

Diagnosis in young children is often a two-stage process.

Stage 1: General developmental screening during well-child checkups

Every child should receive well-child check-ups with a pediatrician or an early childhood health care provider. The American Academy of Pediatrics recommends that all children receive screening for developmental delays at their 9-, 18-, and 24- or 30-month well-child visits, with specific autism screenings at their 18- and 24-month well-child visits. A child may receive additional screening if they have a higher likelihood of ASD or developmental problems. Children with a higher likelihood of ASD include those who have a family member with ASD, show some behaviors that are typical of ASD, have older parents, have certain genetic conditions, or who had a very low birth weight.
Considering caregivers' experiences and concerns is an important part of the screening process for young children. The health care provider may ask questions about the child's behaviors and evaluate those answers in combination with information from ASD screening tools and clinical observations of the child. Read more about screening instruments on the Centers for Disease Control and Prevention (CDC) website.
If a child shows developmental differences in behavior or functioning during this screening process, the health care provider may refer the child for additional evaluation.

Stage 2: Additional diagnostic evaluation

It is important to accurately detect and diagnose children with ASD as early as possible, as this will shed light on their unique strengths and challenges. Early detection also can help caregivers determine which services, educational programs, and behavioral therapies are most likely to be helpful for their child.
A team of health care providers who have experience diagnosing ASD will conduct the diagnostic evaluation. This team may include child neurologists, developmental pediatricians, speech-language pathologists, child psychologists and psychiatrists, educational specialists, and occupational therapists.
The diagnostic evaluation is likely to include:

- Medical and neurological examination
- Assessment of the child's cognitive abilities
- Assessment of the child's language abilities
- Observation of the child's behavior
- An in-depth conversation with the child's caregivers about the child's behavior and development
- Assessment of age-appropriate skills needed to complete daily activities independently, such as eating, dressing, and toileting.
- Because ASD is a complex disorder that sometimes occurs with other illnesses or learning disorders, the comprehensive evaluation may include:
- Blood tests
- Hearing test

The evaluation may lead to a formal diagnosis and recommendations for treatment.

Diagnosis in older children and adolescents

Caregivers and teachers are often the first to recognize ASD symptoms in older children and adolescents who attend school. The school's special education team may perform an initial evaluation and then recommend that a child undergo additional evaluation with their primary health care provider or a health care provider who specializes in ASD.
A child's caregivers may talk with these health care providers about their child's social difficulties, including problems with subtle communication. For example, some children may have problems understanding tone of voice, facial expressions, or body language. Older children and adolescents may have trouble understanding figures of speech, humor, or sarcasm. They also may have trouble forming friendships with peers.

Diagnosis in adults

Diagnosing ASD in adults is often more difficult than diagnosing ASD in children. In adults, some ASD symptoms can overlap with symptoms of other mental health disorders, such as anxiety disorder or attention-deficit/hyperactivity disorder (ADHD). Adults who notice signs of ASD should talk with a health care provider and ask for a referral for an ASD evaluation. Although evaluation for ASD in adults is still being refined, adults may be referred to a neuropsychologist, psychologist, or psychiatrist who has experience with ASD. The expert will ask about:

- Social interaction and communication challenges
- Sensory issues
- Repetitive behaviors
- Restricted interests
- The evaluation also may include a conversation with caregivers or other family members to learn about the person's early developmental history, which can help ensure an accurate diagnosis.

- Receiving a correct diagnosis of ASD as an adult can help a person understand past challenges, identify personal strengths, and find the right kind of help. Studies are underway to determine the types of services and supports that are most helpful for improving the functioning and community integration of autistic transition-age youth and adults.

2.1.5 Treatments and therapies

Treatment for ASD should begin as soon as possible after diagnosis. Early treatment for ASD is important as proper care and services can reduce individuals' difficulties while helping them build on their strengths and learn new skills.

People with ASD may face a wide range of issues, which means that there is no single best treatment for ASD. Working closely with a health care provider is an important part of finding the right combination of treatment and services [1].

2.1.6 Medication

A health care provider may prescribe medication to treat specific symptoms. With medication, a person with ASD may have fewer problems with:

- Irritability
- Aggression
- Repetitive behavior
- Hyperactivity
- Attention problems.
- Anxiety and depression

2.1.7 Behavioral, psychological, and educational interventions

People with ASD may be referred to a health care provider who specializes in providing behavioral, psychological, educational, or skill-building interventions. These programs are often highly structured and intensive, and they may involve caregivers, siblings, and other family members. These programs may help people with ASD:

- Learn social, communication, and language skills.
- Reduce behaviors that interfere with daily functioning.
- Increase or build upon strengths.
- Learn life skills for living independently [1].

2.1.8 ASD statistics

- In 2023, the CDC reported that approximately 1 in 36 children in the U.S. is diagnosed with an autism spectrum disorder (ASD), according to 2020 data.
- Boys are four times more likely to be diagnosed with autism than girls.
- Most children were still being diagnosed after age 4, though autism can be reliably diagnosed as early as age 2.
- 31% of children with ASD have an intellectual disability (intelligence quotient [IQ] <70), 25% are in the borderline range (IQ 71–85), and 44% have IQ scores in the average to above average range (i.e., IQ >85).
- Autism affects all ethnic and socioeconomic groups.
- Minority groups tend to be diagnosed later and less often.
- Early intervention affords the best opportunity to support healthy development and deliver benefits across the lifespan.
- There is no medical detection for autism [2].

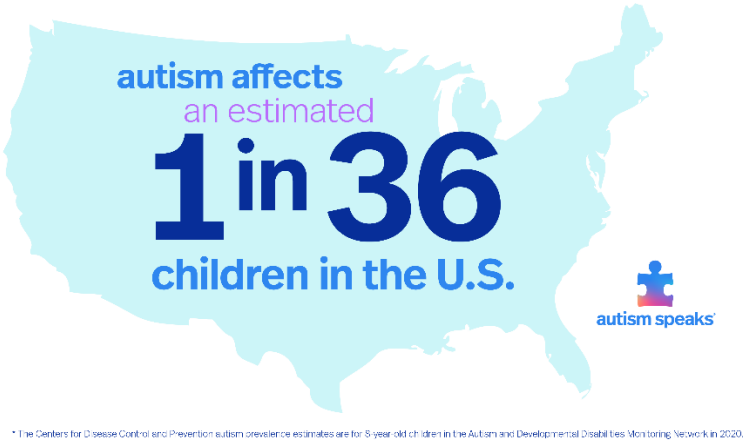


Figure 2: ASD statistics

2.2 Emotional categories

Classification of emotions have a long history and may date as far back as to the first century where a Chinese encyclopedia called the "Book of Rites" described seven feelings, i.e., joy, anger, sadness, fear, love, disliking, and liking. In 1972, Ekman published a list of universal facial emotions (Landowska, 2018 p. 14), consisting of six emotions: anger, disgust, happiness, sadness, fear, and surprise. Others like Robert Plutchik (Rodrigues and Pereira, 2018 p. 850) used eight, which he grouped into four pairs of polar opposites. These are: joy-sadness, anger-fear, trust-distrust, surprise anticipation. However, with Ekmans definitions, it became easier to process emotions when the six universal facial expressions were described more specifically, as:

Anger Lowered eyebrows, glared eyes, tightened lower eyelids, tightened, and narrowed lips, and thrusted jaw.

Fear Raised, pulled together eyebrows, tensed lower eyelids, and a lightly opened mouth.

Disgust Narrowed eyebrows, wrinkled nose, and a curled upper lip.

Happiness Smile and wide eyes

Surprise Raised eyebrows, wrinkled forehead, widely opened eyes, dropped jaw, and an opened mouth.

Sadness Dropped eyelids, lowered corners of the mouth, down casted eyes, and pouted lips [3].

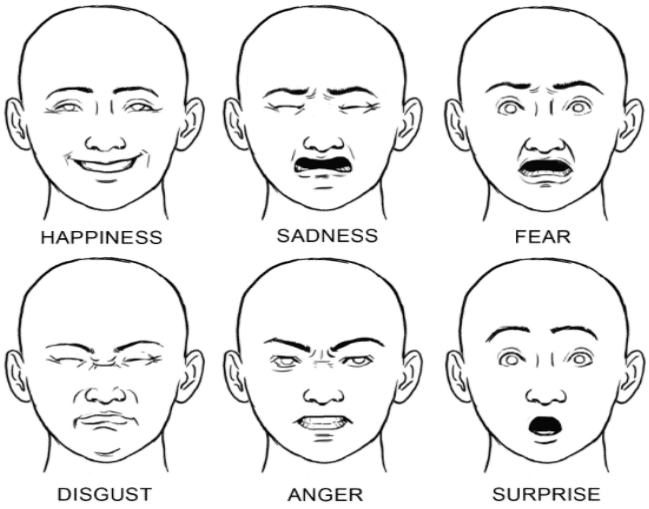


Figure 3. Ekman's 6 Basic Emotions. From Ekman's six Basic Emotions and How They Affect Our Behavior, by Harrison 2021

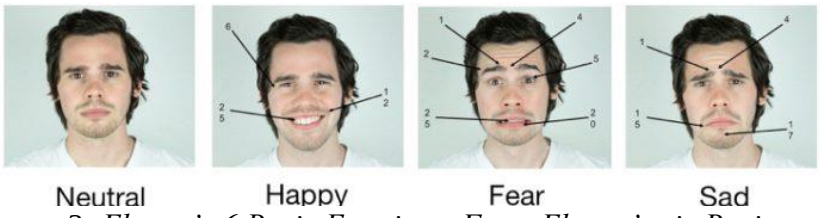


Figure 3: Ekman's 6 Basic Emotions. From Ekman's six Basic Emotions and How They Affect Our Behavior, by Harrison 2021.

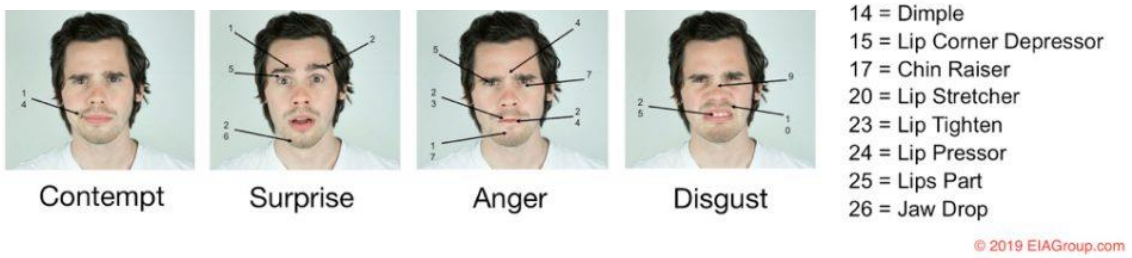


Figure 4: Action Units of facial expressions. From Facial expressions, by Lansley 2022..

Emotion	Action units
Happiness	6+12
Sadness	1+4+15
Surprise	1+2+5B+26
Fear	1+2+4+5+7+20+26
Anger	4+5+7+23
Disgust	9+15+17

Table 1: showing Emotion Action Units

If no specific expression was identified, a neutral expression was added in addition to the six universal facial expressions. Psychologists have over time added several other expressions, like the shown disgust.'

Individual facial muscles and their movements are numbered and called Action Units by FACS. FACS is now a common standard for categorizing physical facial expressions that describe an emotion (Paul Ekman Group LLC, 2020).

By identifying muscle movements by specific numbers or action units, one can digitally represent emotions by concluding on which muscles are activated.E.g.

The intensity of each indicated action can also be graded from weak to strong by characters A to E to arrive at the most likely conclusion for the expressed emotion.

FACS also have additional codes for eye positions, instant behavioral actions like speaking and also head movements like tilting, looking sideways, up/ down or other combinations of these.

The latter classification of head movements can easily be used for augmenting training or test datasets by flipping, mirroring, and rotating original pictures with known expressions. A large increase in samples is beneficial for numerical training in a machine learning algorithm.

From the combined sciences of anatomists and psychologist it is shown that human emotions can be broken down to specific and manageable information items that are suitable for digitization. Once a numerical representation of a facial emotion is obtained, further computerization with numerous algorithms is made possible.

The current work will apply algorithms for the purpose of identifying expressed emotions in a test dataset, and eventually from a random facial picture. For this purpose, so called Artificial Intelligence (AI) is used for identifying muscle movements via deep learning rules for facial expressions, as defined by the FACS standard [3].

2.3 Data processing

This section will look at data processing from the perspective of an image. In most circumstances, many preprocessing steps are required before the dataset can be fed into the network. This is required in order to prepare the data for training.

The network design as well as the input data type must be carefully considered when creating an effective neural network model. The number of images, image height, image width, number of channels, and number of levels per pixel are the most frequent image data input parameters. A colored image has three data channels that correlate to the hues Red, Green, and Blue (RGB), and a greyscale image has one data channel. The most common pixel levels are.

Uniform aspect ratio: The majority of neural network models assume a square-shaped input image, which means that each image must be evaluated for squareness and cropped accordingly. Cropping can be used to choose a square portion of an image.

Image Scaling: Resizing photos to a lesser resolution is a standard process when working with an image-based dataset. This is especially important if the computer lacks a cutting-edge Graphics Processing Unit (GPU), as the model would otherwise be extremely slow. The model will be more effective and more efficient as a result of the down scaling, but some details from the original resolution will be lost. As a result, it’s critical to experiment with various scales to discover the best balance between maintaining enough vital elements and having an effective model.

One Hot Encoding: Another important data pre-processing of images is called one hot encoding. A one hot encoding is a binary vector representation of categorical information. Many machine learning algorithms are unable to operate directly with categorical data. The categories must be numerically transformed.

Normalizing image inputs: Data normalization is a crucial step that assures that each input parameter (in this case, pixel) has a consistent data distribution. This allows for faster convergence while training the network.
By removing the mean from each pixel and dividing the result by the standard deviation, data is normalized. A Gaussian curve centered at zero would be the distribution of such data. We need positive pixel numbers for picture inputs, so we might scale the normalized data in the range [0, 1] or [0, 255].

Dimensionality reduction: The RGB channels could be combined into a single gray-scale channel. When the neural network performance is allowed to be invariant to that dimension, there are typically considerations to lower other dimensions or make the training problem more manageable.
If color has no significance in your images to classify then it's better to go for grey scale images to avoid false classification and complexities [3].

2.4 Artificial intelligence

Artificial Intelligence is an area of computer science that tries to develop programs that have some intelligence, such as performing tasks based on multiple and different or stochastic inputs, with results that resemble reasoning, concluding, or planning. Intelligence also involves learning, classification and storing memories or saved data for future reference by processing e.g., pictorial images. Artificial intelligence is in this context an application of machine learning, or perhaps the application is better described as artificial learning.

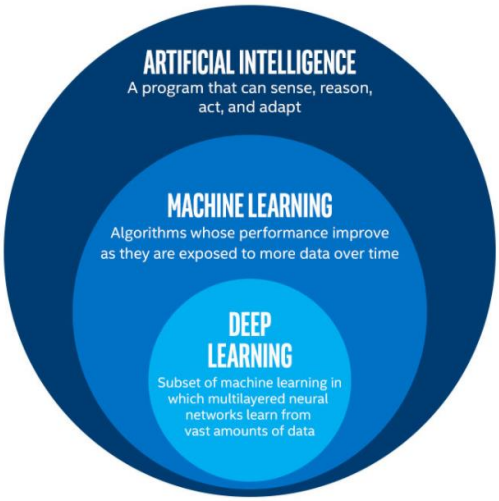


Figure 5: Correlation between AI, ML and DL. From Is machine learning required for deep learning? by Suman 2019.

2.4.1 Deep Learning

Deep Learning is a set of algorithms that tries to resemble the structure of a human brain with its numerous interconnectivities and is thus called an artificial neural network. As opposed to Machine Learning that use sets of input data with known answers for features, deep learning algorithms may extract features from raw pixel data, called feature learning. This is illustrated in Figure below, where DL can skip feature extraction compared to ML. Traditional machine learning approaches require the manual implementation of various computer vision techniques to extract the appropriate features prior to classification. Convolution layers from CNN's are used to extract features from the layers automatically.

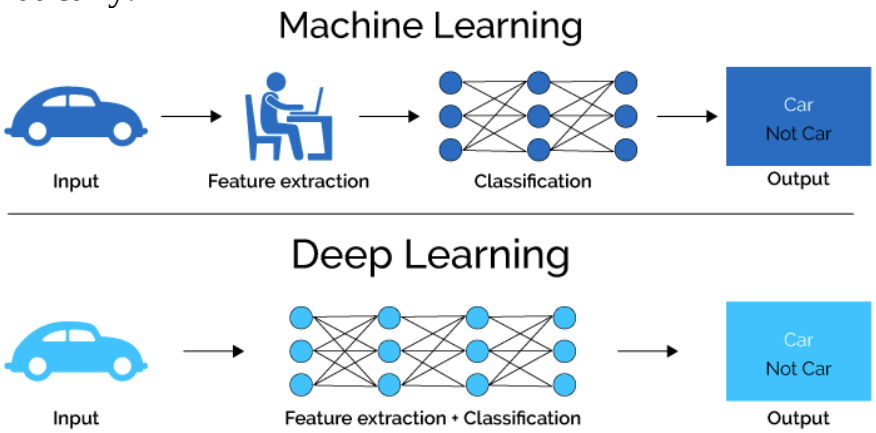


Figure 6: Deep learning vs Machine learning

A deep learning algorithm is tweaked for the task at hand such that results and predictions will improve by training on more and more examples. Parameters are introduced for measuring the accuracy and precision of algorithms.

Benefits of using CNN versus traditional ML

Deep learning algorithms scale source data in different ways. By expanding or augmenting datasets by e.g., flipping, mirroring and rotating original pictures with known expressions, more and more picture patterns or features may match patterns found in new pictures and give correct interpretations.

Augmenting datasets by artificially changing source data in a machine-like procedure, this will increase the capacity to recognize and classify features in new pictures. Classifications are made in stages of different resolutions, in blocks connected as a neural network.

2.4.2 Selecting ML dataset.

Pandas may be used to load data as well as to explore data by statistics and visualizations. UCI datasets have been assembled over a long time and have smaller but some famous datasets like the iris flower dataset from 1936. Dataset assemblies are often referred to as repositories.

OpenML is another and newer repository that can be searched by wanted names, and it has a web API to easily retrieve data.

Kaggle has many large datasets that also can be searched by name. The ImageNet image recognition tasks, where architecture of VGG, Inception and ResNet were trained have datasets of photographic and video-graphic content with a size of more than 160 GB.

As the number of datasets increase, and have different free content, information on availability may be updated in a “List of datasets for machine learning research” on Wikipedia.

Datasets of many GB are normally not downloaded; they are loaded as network libraries, accessed by requests in Python functions, however provided that a good internet connection is available.

Scikit-learn can download dataset data using a built-in real-time API interface.

TensorFlow can also be used for downloading in machine learning projects, however an API needs to be installed separately.
Each dataset has different APIs that may return slight differences in formats such that it may be required that e.g., Scikit-learn must be used to re-format or create the dataset to be processed.

2.4.3 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a class of ANN algorithms. A convolutional neural network uses several blocks of layers during the process of filtering, sorting, and classifying information from the source. These layers may be a convolutional layer, a pooling layer, and a fully connected classifying output layer (Brownlee, 2020) [4].

A convolutional layer is first used to specify the number of filters to be used in the learning process of creating feature maps. In addition, properties of filters are defined, like pixel size used in each filter. A kernel shape is defined [4].

A feature map is a two-dimensional map of activated defined features, with position and strength of a detected feature in an image, in our case a muscle reaction in an emotion [4].

Pooling layers are used to down sample feature maps to more easily identify and sort features from patches of the full original picture, by use of fewer pixels. Calculations may return the maximum, or the most probable feature value from each patch of the full feature map.

Classifier Layer are used after features have been extracted. The feature map is interpreted to arrive at a prediction, in our case an emotion of a face in a photograph. Feature values from each patch can be assembled to a representative full feature map that is interpreted by a classification output layer. The classification layer will typically predict a value or a set of values that is used for identification of an emotion.

The key to create effective convolutional neural networks is to find an architectural design that best utilize these individual layers, and the combination and number of layers. Deep CNN’s have e.g., several convolutional and / or pooling layers [3] [4].

2.4.3.1 CNN Parameters

Some functions and parameters are often encountered in a description of a Convolutional Neural Network and are listed below.

Activation function: The activation functions are located between the network layers and determine which nodes should be fired, that is, nodes that are close to one another.

Optimization function: To minimize the error, this function is used to change weights of a recognition.

Regularization: L1 regularization L2 regularization

Learning rate: This parameter controls how much the weights change between iterations, or how quickly the network evolves.

Batch size: When working with big training sets, it is common to divide them into different batches and train each one in turn. The number of samples in a subset, or batch, is referred to as batch size.

Epochs: An epoch is a complete data cycle. One pass is prediction, cost calculation, and weight update learning cycle [3].

2.4.4 Transfer learning

Normally, an added convolutional layer close to your Baseline Model input layer may learn simple one-dimensional features such as lines. Layers in the middle may combine extracted low-level features and learn more complex features, and layers closer to the output may classify the extracted feature and give weights to different types of recognition. To train and optimize your own model on your own database may be very time-consuming.

The motivation for the transfer learning process was to decrease the requirement for extended training sets. The approach of modifying the weights and parameters of an already trained network to meet your goal is known as transfer learning. It is sometimes referred to as using a pretrained network [3].

Pre trained models

From the keras applications (Keras Documentation, n.d.), we can pick and select any of the state-of-the-art models and use it for our problem.
Pictures from CNN models have a predefined format. When models are loaded into a new Baseline Model from their database, images may be specified to have a new pixel input shape or even be in a grey-scale format to suit training or model weight calculations in your own model.

If, however you would like to benefit from what these architectures already have obtained in feature recognition or in model weight calculations, your own picture format must be be transformed to what these models require:

• The VGG16 Pre-trained Model is expected to get square shaped color images of size 224×224
• The ResNet50 Pre-Trained Model is expected to get square shaped color images of size 224×224 [5].

Table 2: Pre trained models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

2.5 Mobile application

A mobile app (or mobile application) is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers.

Mobile apps are sometimes categorized according to whether they are web-based or native apps, which are created specifically for a given platform. A third category, hybrid apps, combines elements of both native and web apps [6].

2.5.1 Different Types of Mobile Apps

01. Native Apps

Native apps are built specifically for a mobile device's operating system (OS). Technology Used: Native apps are coded using a variety of programming languages. Some examples include: Java, Kotlin, Python, Swift, Objective-C, C++, and React.

Pros:

- Faster and more reliable performance compared to other types of mobile apps.
- Efficient utilization of device resources.
- Utilization of native device UI for an optimized user experience.
- Direct access to a wide range of device features, including Bluetooth, contacts, camera, NFC, and more.

Cons:

- Duplicate development efforts for each platform, increasing costs.
- Inability to reuse code across different platforms.
- Maintenance and updates require managing separate codebases for each version [7].

02. Web Apps

These are known as the Responsive Website Version. It is an application that is accessed via a web browser over a network like internet. It uses a combination of server-side scripts like PHP & asp to handle the storage & retrieval of the information and client-side scripts like JavaScript, CSS, and HTML5 to present information to users.

Technology Used: Web apps are designed using HTML5, CSS, JavaScript, Ruby, and similar programming languages used for web work.

Pros:

- Web Apps run on multiple platforms.
- These are not required to be installed on the device.
- Lower in Cost.

Cons:

- dependent on the browser used on the device, resulting in potential variations in available functionalities and user experiences.

- Web apps, being shells for websites, may not work entirely offline. Even with an offline mode, an internet connection is required for data backup, accessing new data, or refreshing content on the screen [7]

3. Hybrid Apps

A hybrid application is an application that combines the features of both Native Apps & Web Apps. In other words, Hybrid apps are basically web apps that are deployed in a native container. It includes the feature of an embedded browser to enhance access to dynamic content.

Technology Used: Hybrid apps use a mixture of web technologies and native APIs. They're developed using: Ionic, Objective C, Swift, HTML5, and others.

Pros:

- can be deployable across multiple platforms.
- Enhanced User Experience
- Controlled Costs
- Development takes less time.

Cons:

- Hybrid apps might lack in power and speed, which are hallmarks of native apps [7].

2.5.2 Frontend App Development

Frontend, also known as "client-side," is the part of a mobile app, website, or desktop software that the client interacts with directly. It includes all the visual design elements like layout, colors, text fonts, images, and videos, as well as content, interactive elements (buttons, drop-down menus), and navigation. The purpose of the frontend development is to create an eye-pleasing and easily understandable interface that will help users achieve their goals in the app [8].

2.5.3 Backend Development

Backend, or "server-side," is the part of a website or mobile app that is hidden from users. It's responsible for collecting and storing the data as well as pulling the relevant

data upon request. Backend development also handles business logic and manages behind-the-scenes operations like monetary transactions or providing data security [8].

2.5.4 Android operating system

Android is an operating system that is built basically for Mobile phones. It is based on the Linux Kernel and other open-source software and is developed by Google. It is used for touchscreen mobile devices such as smartphones and tablets. The app is developed on an application known as Android Studio. These executable apps are installed through a bundle or package called APK (Android Package Kit).
Android Programming Languages [9].

In Android, programming is done in two languages JAVA or Kotlin and XML (Extension Markup Language). The XML file deals with the design, presentation, layouts, blueprint, etc (as a front-end) while the JAVA or KOTLIN deals with working of buttons, variables, storing, etc (as a back end)
Java or Kotlin?

Kotlin is the official language for Android App Development Declared by Google, and it is the most used language as well. Many of the apps in the Play Store are built with Kotlin and it is also the most supported language by Google. Kotlin is faster and easy than java. Kotlin also include Lot of new features and libraries that is not present in Java [9].

Java is the native language used by Android, applications that helps to communicate with the operating system and hardware that directly uses Java. This language allows the creation of any program and supports almost all types of machines, and OS X be it Android, Windows, or Linux. Java was developed by Sun Microsystems (now the property of Oracle) and one can use Microservices with Java [9].

Kotlin is a cross-platform programming language that can be used as an alternative to Java for Android App Development. It has also been declared “official” language by google. The only sizable difference is that Kotlin removes one of the features of Java such as null pointer exceptions. It also removes the use of semicolon at the end of every line. In short, Kotlin is much simpler as compared to Java [9].

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. XML tags are not predefined in XML. We must define our own Tags. Xml as itself is well readable both by human and machine. Also, it is scalable and simple to develop. In Android we use xml for designing our layouts because xml is lightweight language, so it doesn’t make our layout heavy [10].

Chapter 3

Literature Review

FACIAL EMOTION RECOGNITION USING DEEP LEARNING by Ching-Da Wu and Li-Heng Chen from The University of Texas at Austin used models in the figure(1)a VGG16 based transfer learning model that showed 67.1% accuracy on FER2013 dataset and it implemented a real-time image recognition system including its UI design. The main recognition algorithm is achieved by using data-driven deep learning model along with properly choosing DNN architecture and training procedure. Also, several side algorithms are deployed to increase the smoothness of this system [11].

Architecture	Parameters	Accuracy	Run Time (ms)
VGG16	16296K	**0.671**	1136.473
VGG19	21606K	0.666	1357.670
ResNet50	33034K	0.618	1643.625
DenseNet121	9143K	0.493	1921.401
DenseNet169	15404K	0.292	2953.563
MobileNet	5335K	0.479	**603.593**
MobileNetV2	8558K	0.533	782.538
From Scratch	N/A	0.648	684.758

Table 3:Results of FACIAL EMOTION RECOGNITION USING DEEP LEARNING

In Facial Emotion Recognition(State of the Art Performance on FER2013) Since 2021 achieves single-network state-of-the-art classification accuracy on FER2013 using a VGG16. They thoroughly tune all hyperparameters towards an optimized model for facial emotion recognition. Different optimizers and learning rate schedulers are explored, and the best initial testing classification accuracy achieved is 73.06 %. They also carry out extra tuning on our model using Cosine Annealing and combine the training and validation datasets to further improve the classification accuracy to 73.28 %. For future work, we plan to explore different image processing techniques on FER2013 and investigate ensembles of different deep learning architectures to further improve our performance in facial emotion recognition [12].

The paper ResNet-50 and VGG-16 for recognizing Facial Emotions by Dhankhar from 2019, explored the VGG-16 and ResNet50 architectures for recognizing facial emotions using deep learning. They also developed a Support Vector Machine (SVM) classifier baseline model for comparison purposes. The article used both the () dataset and the Karolinska Directed Emotional Faces (KDEF) dataset for testing of their two models. In the introduction they state that they expect their best model to achieve at least 60% test set validation because the winner of Kaggle’s Facial

Expression Recognition Challenge achieved 71.2% accuracy and the top ten contestants achieved at least 60% accuracy [13].
On the KDEF dataset, overall results on accuracies, precision, and recall with their models were higher than on the dataset. Surprisingly, all four models gave better results on the KDEF dataset compared to the dataset, which is a much smaller dataset. However, the photos in the KDEF data collection are of greater quality and the dataset contained examples of text overlay in the image's background. Below, the results on the KDEF dataset will be shown and what results application of transfer learning could give. The results of the SVM (baseline), VGG-16, ResNet50 on the KDEF dataset are shown in Table 3.1 below.

	Accuracy	Precision	Recall
SVM (baseline)	37.9 %	50.1 %	54.9 %
VGG-16	71.4 %	81.9 %	79.4%
ResNet50	73.8 %	83.3 %	80.7%

Table 4: KDEF dataset performance. From ResNet-50 and VGG-16 for recognizing Facial Emotions by Dhankhar 2019.

Applying transfer learning further improved the results. The results after transfer learning was implemented is shown in Table 3.2 below.

	Accuracy	Precision	Recall
VGG-16	73.6%	84.2%	81.1%
ResNet50	76.0%	86.1%	82.5%

Table 5: KDEF dataset performance with Transfer Learning. From ResNet-50 and VGG-16 for recognizing Facial Emotions by Dhankhar 2019.

The findings were even better when transfer learning was used. They fixed the layer weights of the pre-trained models except for the last few layers and retrained them on the KDEF dataset. Resnet-50 improved from 73.8% to 76.0%. Precision and recall both improved in the same way. This demonstrates that their model was able to take learnings from datasets and apply them to the KDEF dataset [13].

Facial Emotion Recognition using Deep Learning BY Emilia Basioli Kirkvik This paper discusses facial expression recognition (FER) technology, which utilizes machine learning (ML) to analyze facial expressions and determine a person's emotional state. The paper focuses on deep learning (DL) models, particularly convolutional neural networks (CNNs), which have shown promising results due to their automatic feature

extraction and computational efficiency. The paper compares and evaluates three DL models: a sequential CNN model, a pre-trained Resnet50 model, and a pre-trained VGG16 model, using a Facial Expression Recognition 2013 (FER-2013) dataset. The sequential CNN model achieved an accuracy of 89.7%, while the two pre-trained models achieved an accuracy of 86.5% without fine-tuning. The paper suggests that transfer learning can be used to obtain similar accuracy with smaller models and recommends investigating other pre-trained models on more diverse datasets. The paper concludes that the DL models performed well and converged towards better values, but some misinterpretations on unseen datasets were expected [14].

Literature	Number of facial expressions	Dataset	Best accuracy
[11]	7	FER2013	VGG16 67.1%
[12]	7	FER2013	VGGNet, (CNN) architecture 73.06 %,
[13]	7	KDEF	ResNet50 76.0%
[14]	7	FER2013	CNN 89.7%

Table 6: PREVIOUS WORK CLASSIFICATION AND ACCURACY

Chapter 4

Methodology

4.1 Block diagram

Based on the problem at hand, it seems that there are three main sections that need to be addressed in order to solve it.

The first section involves developing a software model that can accurately predict emotions based on facial expressions. This will require choosing an appropriate deep learning architecture, training it on a suitable dataset, and fine-tuning it to achieve high accuracy.

The second section involves developing an application that will serve as the user interface. This application will need to be intuitive and user-friendly, with clear instructions and feedback for the user. It should allow the user to easily interact with the software model and receive real-time feedback on their emotional state.

The third section involves integrating a camera into a pair of glasses that can capture images. This camera will need to be high-quality and capable of capturing clear images of the user's facial expressions in a variety of lighting conditions. It will also need to be connected to the software model and the user interface application in order to provide real-time feedback on the user's emotional state.

Overall, the key challenge will be to ensure that all three sections work seamlessly together, so that the user can easily and accurately receive feedback on their emotional state in real-time. This will require careful coordination and testing of each component, as well as ongoing monitoring and refinement to ensure that the system is working optimally.
That’s all described in the figure below.

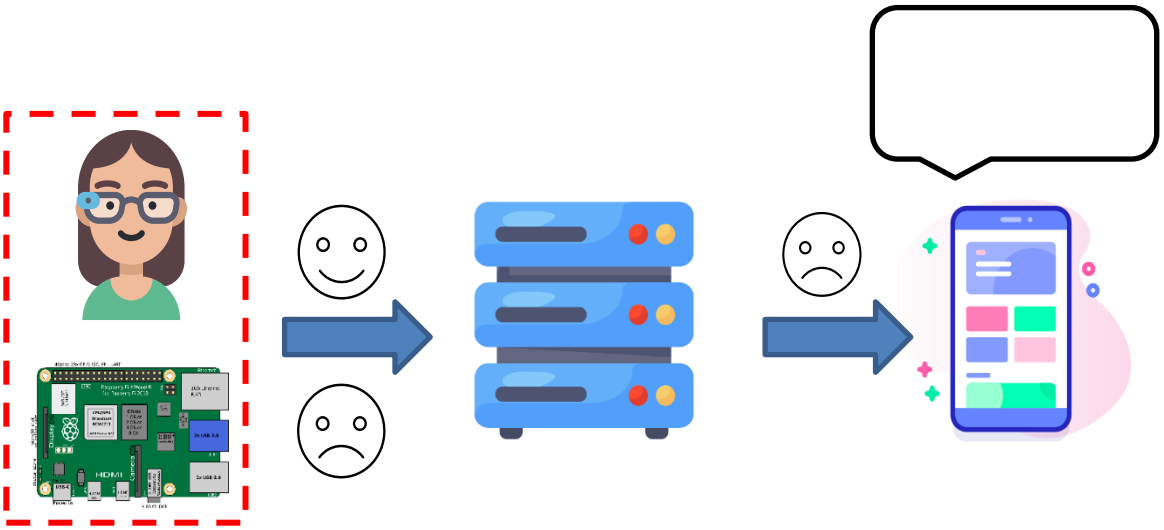


Figure 7: Block diagram of overall system

4.2 Dataset

We shall introduce a few datasets in this part. The datasets are chosen to be as large and diversified as feasible in terms of quantity and size.

4.2.1 CK+ Dataset

Cohn-Kanade Dataset (CK+) that contains 920 individual facial expressions.
It contains adapted data up to 920 images from 920 original CK+ dataset.
Data is already reshaped to 48x48 pixels, in grayscale format and face cropped using haarcascade frontalface default.
Noisy (based on room light/hair format/skin colour) images were adapted to be clearly identified using Haar classifier.
Columns from file are defined as emotion/pixels/Usage.
Emotions are defined as determined index below:

0 : Anger (45 samples)
1 : Disgust (59 samples)
2 : Fear (25 samples)
3 : Happiness (69 samples)
4 : Sadness (28 samples)
5 : Surprise (83 samples)
6 : Neutral (593 samples)
7 : Contempt (18 samples)
Pixels contains **2304 pixels** (48x48) each row.

Figure 8: CK+ Dataset

Cons:

- Small number of images per emotion: While the CK+ dataset contains a large number of images overall, the number of images per emotion is relatively small. This can make it challenging to train models that are highly accurate for all emotions.
- Limited diversity: The dataset was collected under controlled laboratory conditions, which may not fully capture the diverse range of facial expressions that occur in real-world settings.
- Imbalanced classes: The number of images per emotion is not evenly distributed, with some emotions having significantly fewer images than others. This can make it challenging to train models that are equally accurate for all emotions.
- Noisy images: The dataset may contain some noisy images due to variations in lighting, hair format, and skin color. This can make it more difficult to accurately identify and label emotions for these images [15].

4.2.2 AffectNet Dataset

AffectNet is a large-scale dataset for facial expression recognition that contains over one million images with corresponding emotion labels. The dataset was created by researchers at the University of Pittsburgh and was made publicly available in 2017.

The images in AffectNet were collected from a variety of sources, including online image search engines, social media, and public photo-sharing websites. The images are labeled with one of eight emotion categories: Neutral, Happy, Sad, Surprise, Fear, Disgust, Anger, and Contempt. In addition to emotion labels, the dataset also includes annotations for facial landmarks and attributes such as age, gender, and ethnicity.

AffectNet is a large and diverse dataset that includes images from a wide range of ages, genders, and ethnicities. The images were collected under various conditions, including different lighting, backgrounds, and poses. This makes AffectNet well-suited for training and evaluating facial expression recognition models in real-world scenarios.

One potential limitation of AffectNet is that the emotion labels were assigned by human annotators, which can introduce some level of subjectivity and inconsistency. However, the dataset includes multiple annotations per image to help mitigate this issue. Additionally, the dataset includes a validation set with ground-truth labels, which can be used to evaluate the performance of models trained on the training set.

Cons:

- One potential limitation of the AffectNet dataset related to its size is the potential computational cost associated with using such a large dataset. Training machine learning models on a dataset with over one million images can be computationally expensive and time-consuming, particularly for researchers who may not have access to specialized hardware or high-performance computing resources. Additionally, working with such a large dataset can require significant storage space and memory.
- Another potential limitation related to the size of AffectNet is the potential for class imbalance. While the dataset includes a large number of images overall, the number of images per emotion is not evenly distributed, with some emotions having significantly fewer images than others. This can make it challenging to train models that are equally accurate for all emotions and may require additional techniques such as data augmentation or class weighting to address this issue.

Finally, while the large size of AffectNet can be a pro in terms of providing a diverse and comprehensive dataset for facial expression recognition, it can also be a con in terms of the potential for overfitting [16].

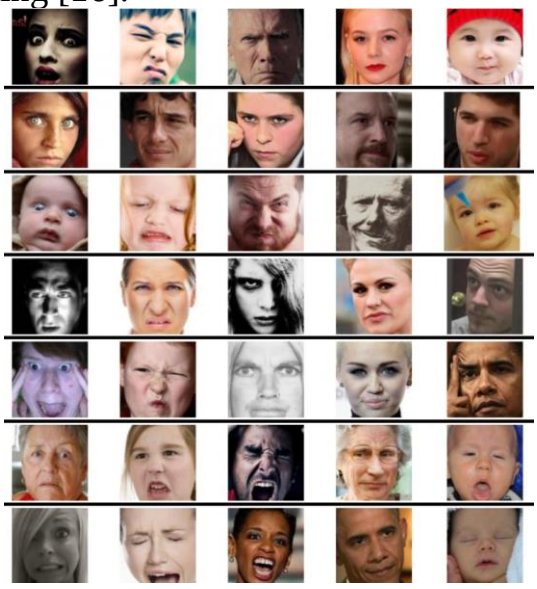


Figure 9: AffectNet Dataset

4.2.3 Dataset description (FER2013) *chosen dataset*

The Facial Expression Recognition 2013 (FER2013) dataset is a publicly available dataset that is commonly used for training and testing facial expression recognition models. It consists of 35,887 grayscale images, each of size 48x48 pixels, representing seven different facial expressions: anger, disgust, fear, happiness, sadness, surprise, and neutral.

The dataset is divided into three sets: a training set of 28,709 images, a public test set of 3,589 images, and a private test set of 3,589 images. The training set is used for training and optimizing the model, while the public and private test sets are used for evaluating the model's performance. The images were collected from a variety of sources, including the internet and professional databases, and are labeled with the corresponding emotion category [17].

Figure 10: FER2013 dataset

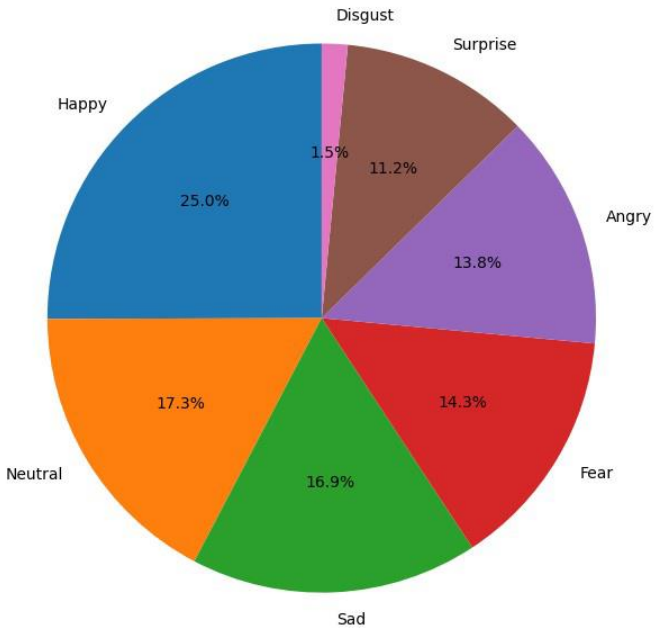


Figure 11: distribution for each of the seven emotions followed by a percentage.

4.3 Pre-processing

- In data preprocessing, the goal is to prepare the dataset for the model. One common issue with datasets is **imbalanced class distribution**, where.

❑ The smallest class is disgust (547 images)
❑ The largest class is happy (8989)

To address this issue, various techniques can be used to balance the dataset:

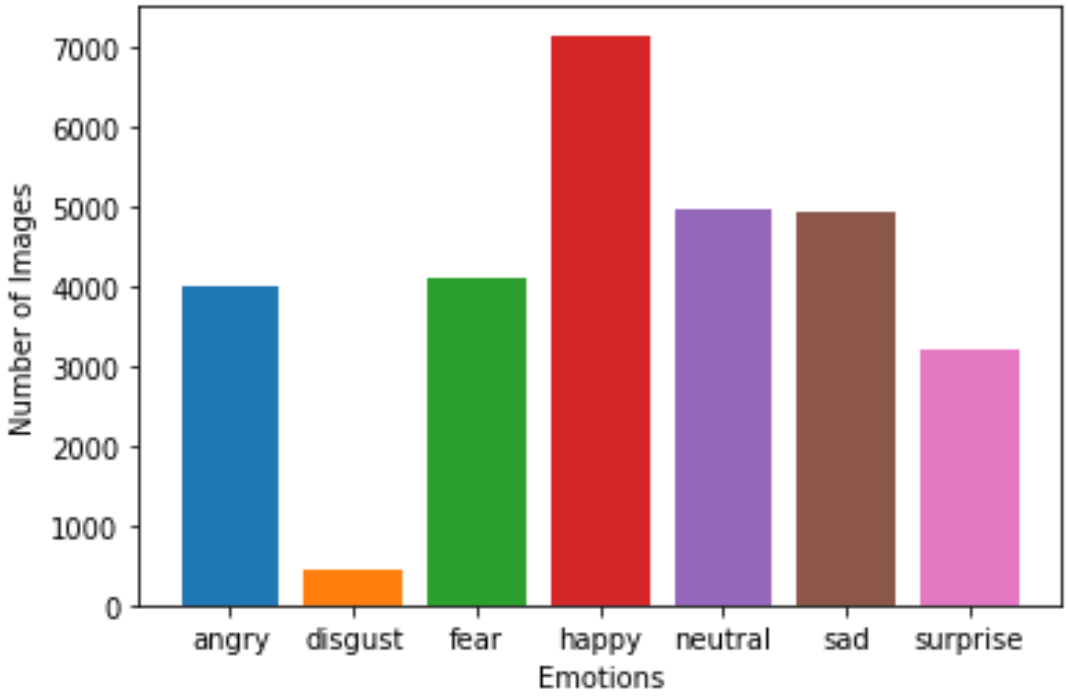


Figure 12: distribution for FER2013 dataset before balancing.

4.3.1 Balancing techniques

2. GAN (Generative Adversarial Network):

It is an algorithm that uses two neural networks (Generator and Discriminator networks) which compete, resulting in instances of data that are fake but look like the original data. To have a full grasp of GAN, understanding how the generator and discriminator work is vital.

GENERATOR

The goal of the generator is to *generate* or produce new and synthetic samples of a certain input, which could be a random set of values or noise. For instance, if you trained it with the class of a cat, the generator would perform a series of computations, and then produce an image of a cat, which isn't real but looks real.

Ideally, the output won't be the same cat for every run, and to make sure the generator produces different samples each time, the input would be a random set of values, known as the noise vector.

Considering the fact that it is 'competing' with the discriminator, the generator does its best to produce a new fake image with the hope that the discriminator would consider the image to be authentic [18].

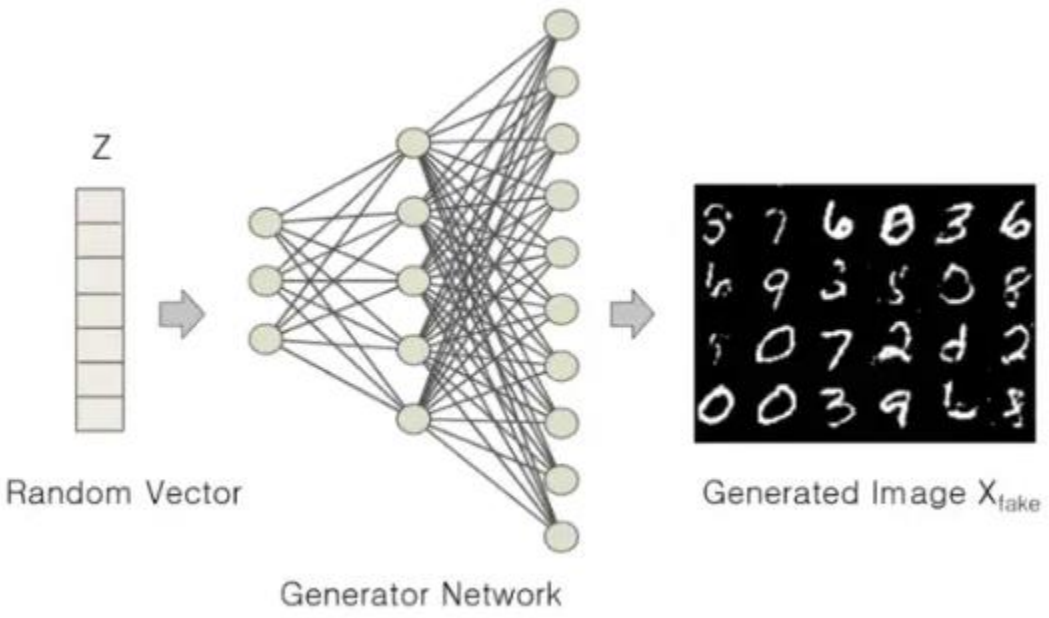


Figure 13: GENERATOR

DISCRIMINATOR

The goal of the discriminator, therefore, is to process the images from the generator and classify them as either real or fake. It works as a binary classifier by taking two inputs: the first being a real image (from training data), and the other being the image the generator produced [18].

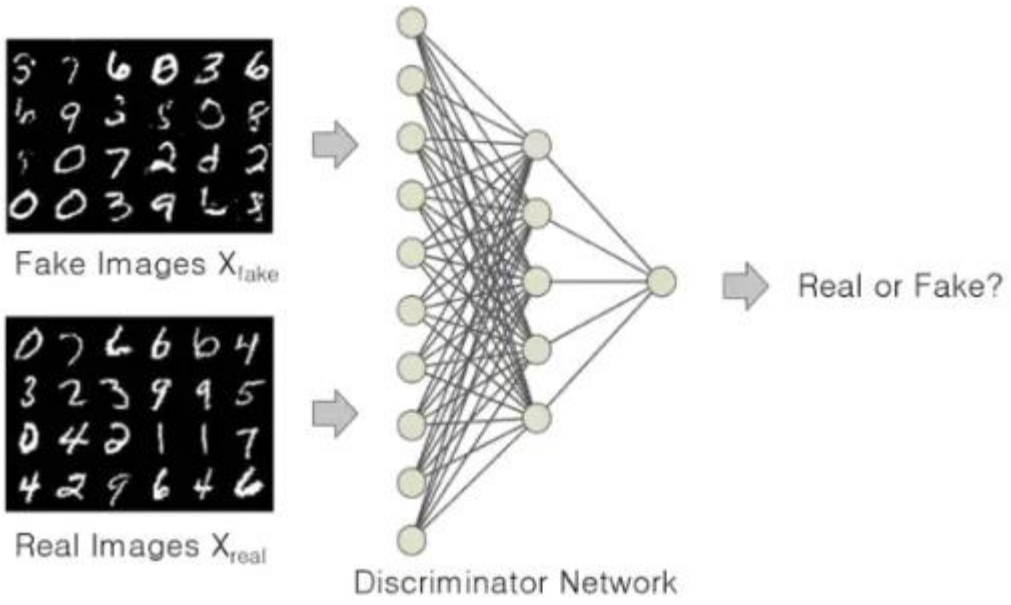


Figure 14: DISCRIMINATOR

In the context of using GANs to generate synthetic samples for a deep learning model, the first trial resulted in images with too much noise.

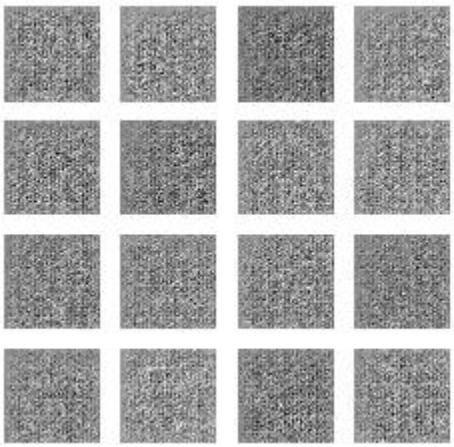


Figure 15: GANs the first trial

In the second trial, the generated images were unrealistic, meaning that they did not resemble the real images in the dataset.

Figure 16: GANs the second trial

In both cases, the quality of the generated images was not sufficient to be used for training the machine learning model. This could negatively impact the performance of the model, as the generated samples may not accurately reflect the underlying patterns in the data.

2. SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE (Synthetic Minority Over-sampling Technique) is a technique for addressing the issue of imbalanced class distribution in a dataset. It involves generating synthetic samples of the minority class by interpolating between existing samples.
The basic idea behind SMOTE is to identify the minority class samples that are close to one another, and then create new synthetic samples along the line segments connecting these samples. This helps to increase the diversity of the minority class samples and balance the dataset.

After trying SMOTE on a dataset, it was observed that the performance of the machine learning model decreased, resulting in lower accuracy. As a result, SMOTE was not used as a technique to address the issue of imbalanced class distribution in the dataset [19].

3. Up sampling

After attempting to up sample the minority class (disgust class) it was observed that the performance of the model did not improve or did not yield good results. As a result, up sampling the minority class was deemed an ineffective solution to address the issue of imbalanced class distribution in the dataset.

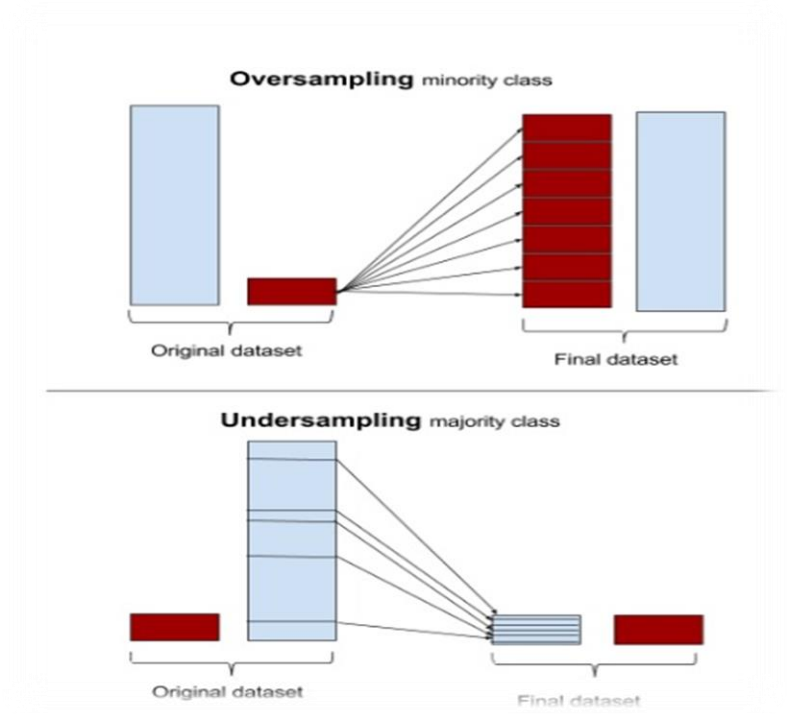


Figure 17: upsampling

4. Under sampling (*The proposed method*)

It would decrease the proportion of majority class until the number is similar to the minority class.

- So, we have down sampled all the classes in your dataset to the size of the lowest class (disgust, which has 547 samples).

4.3.2 Data augmentation

It is a set of techniques used to increase the amount of data in a machine learning model by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It helps smooth out the machine learning model and reduce the overfitting of data [20].

Techniques

Images are modified slightly and then added to the data sets used in machine learning models. Some techniques used to augment images for machine learning algorithm datasets are:

- Geometric transformations
- Elastic transformations
- Flipping
- Color modification
- Cropping
- Rotation
- Translation (moving the image in the x or y direction)
- Noise injection
- Zooming and scaling.
- Random erasing

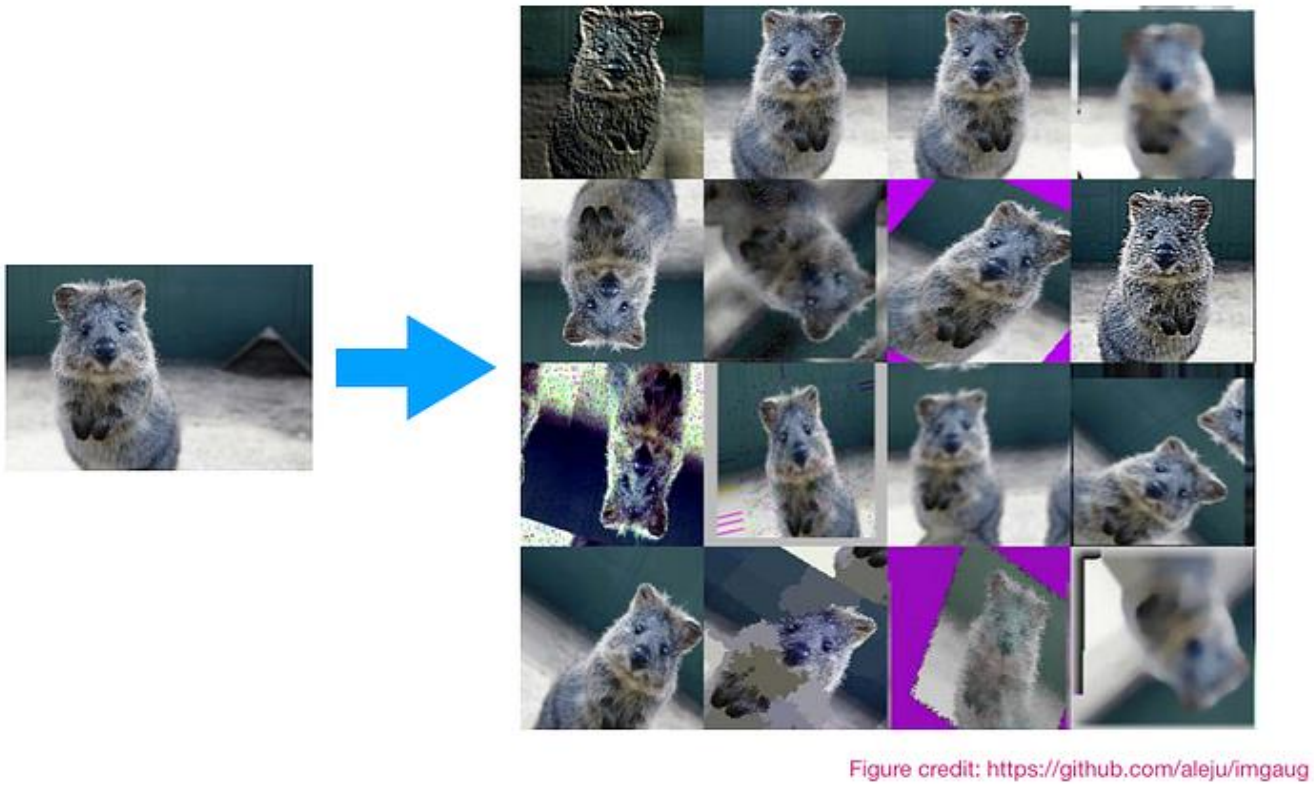


Figure 18: Data augmentation techniques

Benefits of Data Augmentation

A deep learning model performs better and is more accurate when the dataset is rich and comprehensive. By creating fresh and varied instances to train datasets, data augmentation can help improve the performance and results of machine learning models.

Data collection and labeling can be time-consuming and costly for machine learning models. Companies can lower these operational costs by transforming datasets using data augmentation techniques.

Cleaning data is one of the phases required in creating a data model with a high accuracy level. However, if data cleaning reduces representability, the model will not make accurate predictions for real-world inputs. Machine learning models can be made more robust via data augmentation approaches, which create several variances that the model might encounter in the actual world [20].

- To further balance the dataset, we have then up sampled all classes to a size of 2500 using **data augmentation** techniques such as:

(1) intensity variation (-1:2) and
(2) flipping (-1:1). This involves creating new samples by applying these transformations to the existing images in the dataset.

- By using the same data augmentation techniques across all classes, we are ensuring that the changes made to the data are consistent and fair for all classes. This can help to ensure consistency.

This approach helps to address the issue of imbalanced class distribution in the dataset and can improve the performance of the model and increase the size and diversity of the dataset through data augmentation.

.

Figure 19: augmentation techniques

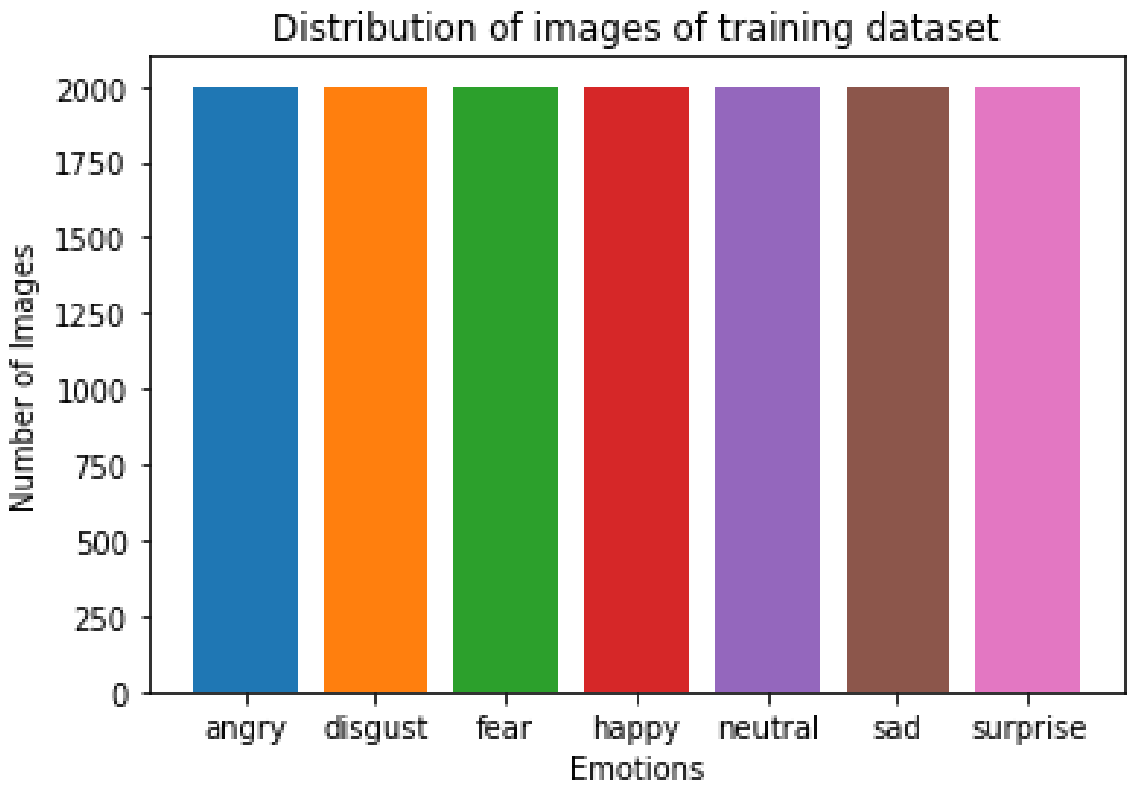


Figure 20: distribution of dataset after balancing

4.3.3 Preparing data for model after balancing

1.Data splitting

In the initial data processing stage, the data was divided into three separate sets: a training set, a testing set, and a validation set, using **an 80-10-10 split**. The purpose of this division was to allow for effective evaluation of the performance of a machine learning model and to prevent overfitting.

The training set, comprising 80% of the initial dataset, was used to train the model and adjust its weights based on the input data and associated labels. The testing set, comprising 10% of the dataset, was used to evaluate the model's performance on previously unseen data, to test the model's ability to generalize. Finally, the validation set, also comprising 10% of the dataset, was used to evaluate the model's performance during training, and to adjust hyperparameters to prevent overfitting.

2. Image to array conversion

The Second part of data processing is to convert the images to nparrays.

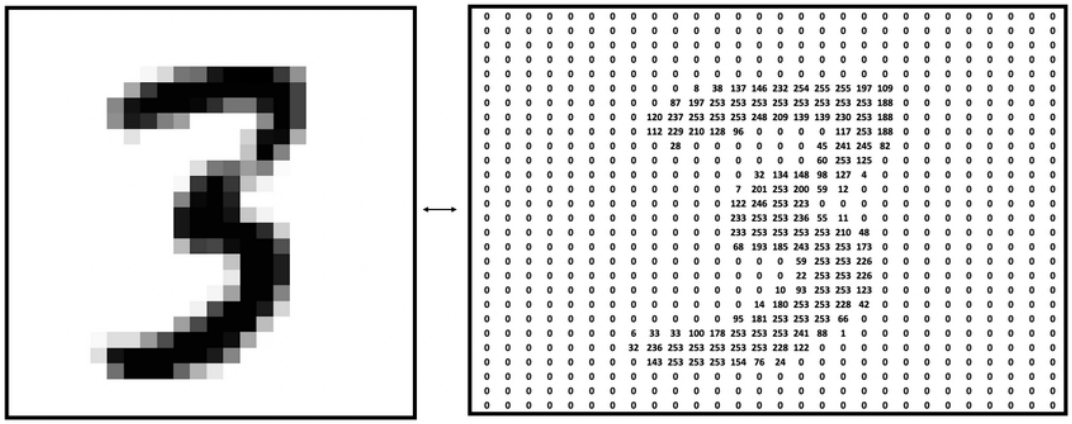


Figure 21: Image to array conversion

"Image to array" is a process of converting image data into a numerical format that can be processed by a machine learning model. This process involves reading an image file and converting it into a multidimensional array of pixel values.

The pixel values in the array represent the intensity of the color at each pixel in the image. These values can be represented as integers or floating-point numbers, depending on the specific requirements of the machine learning model.

The image to array conversion process is an essential step in preparing image data for use in deep learning models. Once the images are converted to arrays, they can be preprocessed and used as input to a machine learning model [21].

3. Normalization

It is a data preparation technique that is frequently used in machine learning. The process of transforming the columns in a dataset to the same scale is referred to as normalization [22].

Next, the images are normalized by the pixel values for colors, by rescaling 256 colors to be represented by range of [0,1].

4.4. Building Sequential CNN model

A Baseline convolutional neural network Model will be established as a typical classification model since emotions are divided into classes.

Since all reviewed literature studies confirm that transfer learning give a clear improvement in predictions for base sequential models, an alternative would then to be investigated how much a complicated baseline model can be reduced to obtain the same accuracy as a less advanced model that uses transfer learning.

By building and analyzing a rather complicated baseline model and comparing results to a less advanced model that uses transfer learning, it will be checked if transfer learning can be used to obtain the same accuracy with significantly smaller models. Transfer learning can then be used as a 'performance optimizer' on smaller baseline models that give sufficiently accurate results with a minimum use of resources [11].

First, a rather complex baseline sequential model is built. Transfer Learning on a smaller model will later be used in comparisons with this model, and results will show how much benefit this gave to a new and modified model. Transfer Learning from several architectures can be used but VGG and ResNet and others will be used.
The baseline model built from scratch is illustrated below.

The model is called Sequential due to its sequence of convolutions and pooling layers. The CNN network structure is defined as described below.

We use a CNN with four convolutional blocks as described in Fig. Each convolutional block consists of a batch normalization, a convolutional layer with (N*N) kernel size with a rectifier-linear-unit (ReLU) activation function, and a max pooling layer with (2*2) size. Batch normalization is utilized to ensure the input data have zero mean and unit variance. All the convolution layers have (1*1) stride. The four convolution layers have {64; 128; 512; 512} kernels, respectively.
Following the four convolution blocks are three fully connected layers with sigmoid activation and output sizes of {256; 256; 7}. Dropout layers are placed before each of the three fully connected layers with dropping probabilities 0:3 [11].

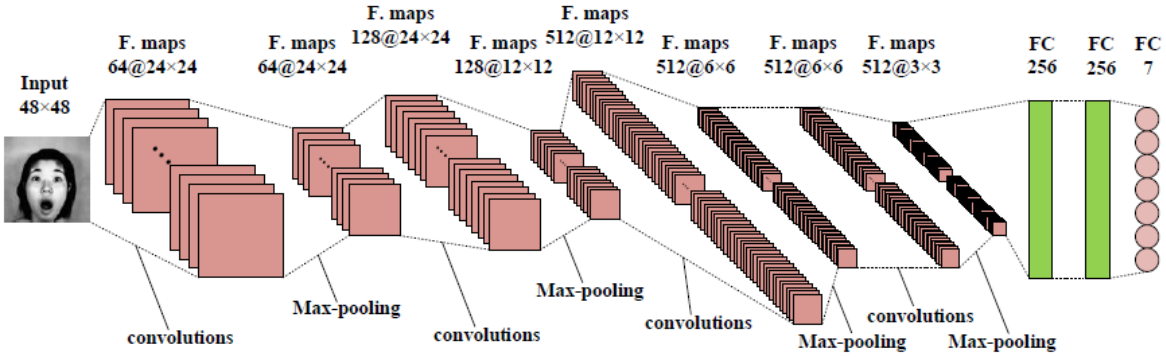
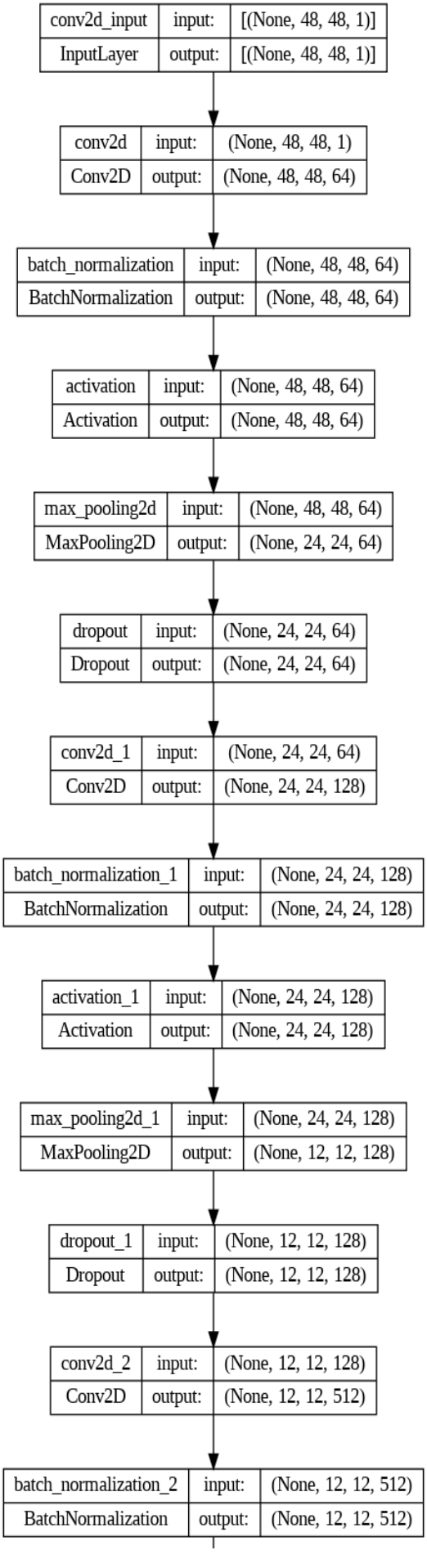


Figure 22 : Illustration of our 4-layer built-from-scratch DNN model. Each layer is composed of convolutions, max-pooling, and a Rectified Linear Unit (ReLU) activation function. The parameterization of the convolutional layer is denoted as" Channels @ width _ height".

The model is later fitted with the train dataset, using a batch size of 32 and with 50 epochs.

Better end results will normally be obtained with more epochs. A run with up to 50 epochs was at one time tested to see if the rate of convergence improved significantly. If there is no significant improvement, fewer epochs are used.

But we Use the same number of epochs, such as 50 to compare the performance of a scratch CNN and a transfer learning model.


conv2d_input | input: | [(None, 48, 48, 1)]
InputLayer | output: | [(None, 48, 48, 1)]
conv2d | input: | (None, 48, 48, 1)
Conv2D | output: | (None, 48, 48, 64)
batch_normalization | input: | (None, 48, 48, 64)
BatchNormalization | output: | (None, 48, 48, 64)
activation | input: | (None, 48, 48, 64)
Activation | output: | (None, 48, 48, 64)
max_pooling2d | input: | (None, 48, 48, 64)
MaxPooling2D | output: | (None, 24, 24, 64)
dropout | input: | (None, 24, 24, 64)
Dropout | output: | (None, 24, 24, 64)
conv2d_1 | input: | (None, 24, 24, 64)
Conv2D | output: | (None, 24, 24, 128)
batch_normalization_1 | input: | (None, 24, 24, 128)
BatchNormalization | output: | (None, 24, 24, 128)
activation_1 | input: | (None, 24, 24, 128)
Activation | output: | (None, 24, 24, 128)
max_pooling2d_1 | input: | (None, 24, 24, 128)
MaxPooling2D | output: | (None, 12, 12, 128)
dropout_1 | input: | (None, 12, 12, 128)
Dropout | output: | (None, 12, 12, 128)
conv2d_2 | input: | (None, 12, 12, 128)
Conv2D | output: | (None, 12, 12, 512)
batch_normalization_2 | input: | (None, 12, 12, 512)
BatchNormalization | output: | (None, 12, 12, 512)

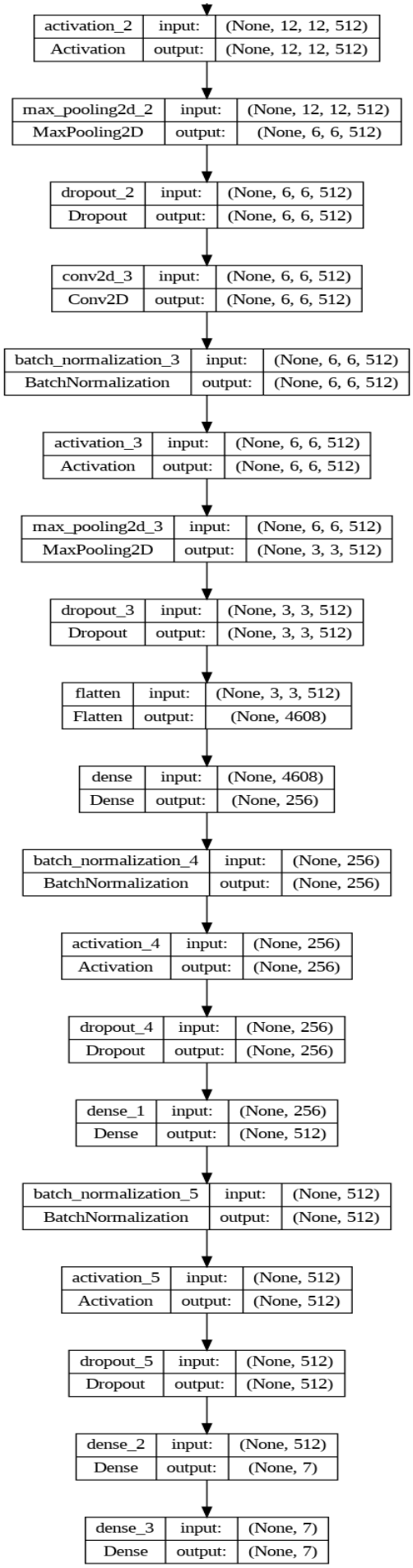


Figure 23: Illustration of the CNN Sequential model

4.5 Building transfer learning models

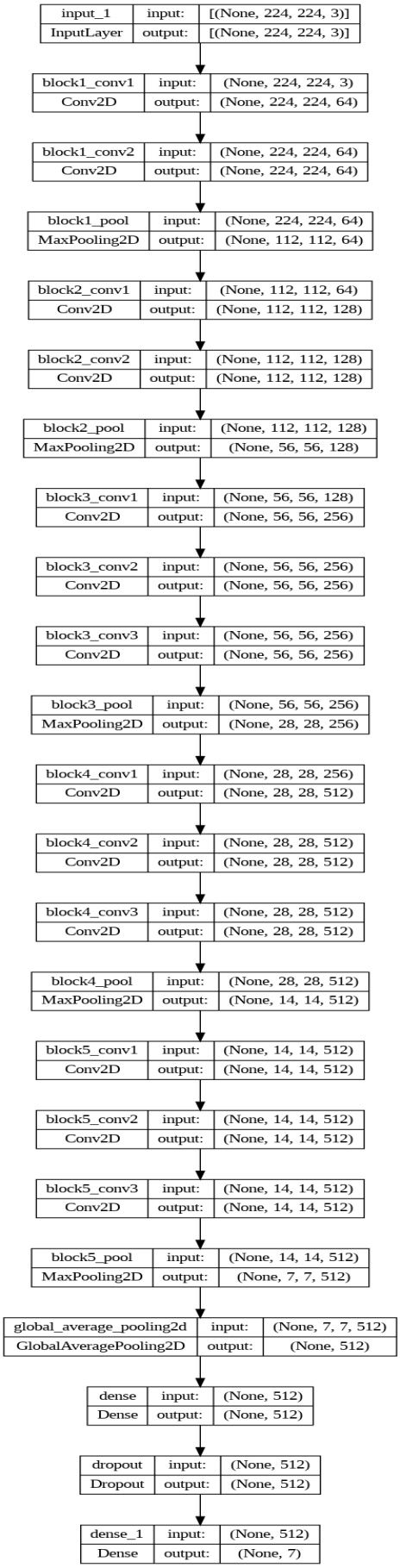
1. Building VGG16 model

The model’s architecture cannot be changed because the weights have been trained for a specific input configuration. This is why a property of the image must be changed from 1 channel grey scale to a 3 channel rgb image.

The ImageNet collection is known to contain photos of various vehicles (sports cars, pick-up trucks, minivans, etc.). We can still import a model that has been pre-trained on the ImageNet dataset and extract features using its pre-trained layers. We can however not use the complete architecture of the pre-trained model. Our Target Dataset has only seven classes for prediction, while the Fully Connected layer generates 1,000 possible for output labels. So, we will take a pre-trained model like VGG16 and "chop off" the Fully Connected layer (also known as the "top" model) [25].

And we changed the input image size to 224, as in VGG16, the input image size is typically set to 224x224 pixels. This means that the input layer of the network is configured to expect input images of size 224x224x3 (i.e., 224 pixels wide, 224 pixels high, and 3 color channels - red, green, and blue).

Figure 24: The pre-trained VGG16 model with weights 'imagenet' illustration of each layer. The trainable layers are freezed, and one Flatten, Dense (512) and Dense(7) layers are added at the end.



2. Building ResNet50 model

This model has undergone pre-training on the ImageNet dataset, utilizing the "imagenet" weight. Due to the specific input configuration used during training, the architecture of the model remains unchanged. Consequently, any alteration in the image properties, such as transitioning from a 1-channel grayscale to a 3-channel RGB image, necessitates a corresponding adjustment [24].

To adapt the ResNet50 model for a different task, we adopt a strategy known as freezing the pre-trained layers. This entails preventing any modifications to these layers during the training process. This approach helps to retain the valuable learned features and enhance the overall performance of the model. Additionally, we introduce a new top layer to the model, which possesses the desired number of output units for classifying our target classes. In our case, as we aim to classify among only 7 classes, we incorporate a new top layer comprising 7 output units.

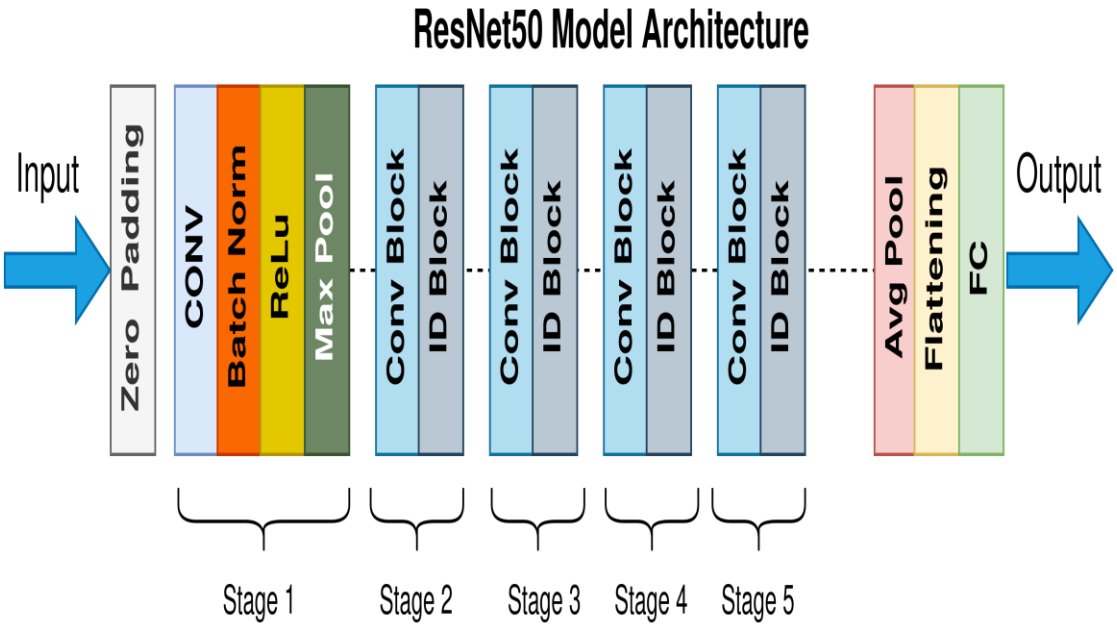


Figure 25: ResNet50 model architecture

3. Building DenseNet201 model

The DenseNet201 model was originally trained on the ImageNet dataset, which consists of RGB images of size 224x224 pixels. Therefore, when using DenseNet201 for transfer

learning, it is recommended to use input images of the same size (224x224 pixels) to ensure compatibility with the pre-trained weights. This is why a change of the property of the image from 1 channel grey scale to a 3 channel RGB image must be done. also, we will change image size into 224x224 pixels [23].

The DenseNet201 model is fine-tuned by freezing all layers except the top layer for classification." and a new top layer with 7 output units is added to classify inputs among 7 specific classes.

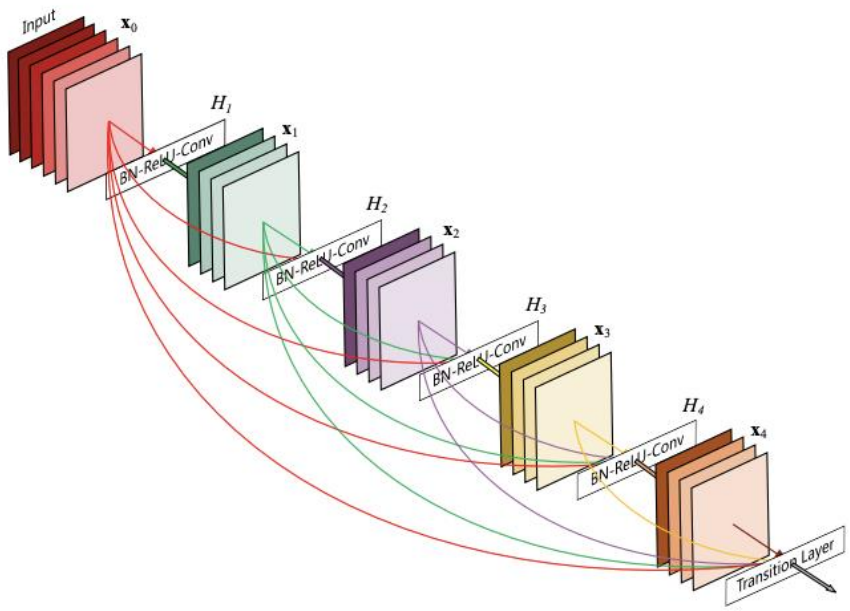


Figure 26: DenseNet201 model architecture

4. Building MobileNet model architecture

The MobileNet architecture is optimized for mobile and embedded devices, prioritizing lightweight and efficient performance. The original MobileNet paper utilized input images of size 224x224 pixels, a commonly adopted choice in convolutional neural networks. Therefore, to align with this standard, a modification is required to transform the image from 1-channel grayscale to a 3-channel RGB representation. Additionally, the image size is adjusted to 224x224 pixels to adhere to the MobileNet architecture's requirements.
In the fine-tuning process, all layers of the model are frozen, except for the top layer responsible for classification. This ensures that the previously learned features are preserved and remain unchanged during training. To address our specific task, a new top

layer with 7 output units is introduced. This top layer enables the model to classify inputs among the 7 designated classes, providing accurate and targeted predictions [26].

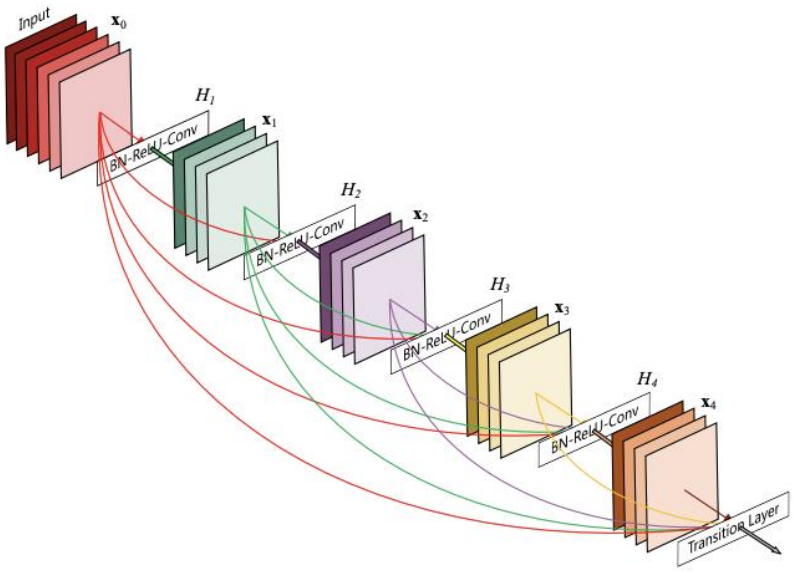


Figure 27: MobileNet model architecture

5. Building EfficientNetB0 and EfficientNetB7 model

EfficientNet, first introduced in Tan and Le, 2019 is among the most efficient models (i.e. requiring least FLOPS for inference) that reaches State-of-the-Art accuracy on both imagenet and common image classification transfer learning tasks. We will be using the **EfficientNetB0** and **EfficientNetB7** architecture.

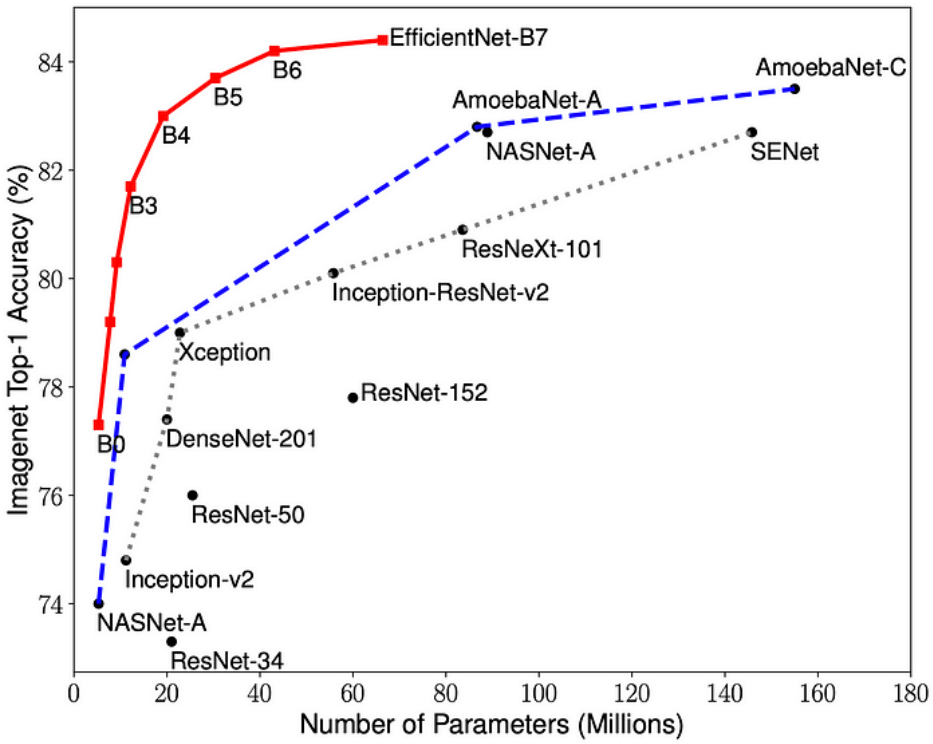


Figure 28 The EfficientNet family compared to other ImageNet models.

For B0 to B7 base models, the input shapes are different. Here is a list of input shape expected for each model [27]:

Base model	resolution
EfficientNetB0	224
EfficientNetB1	240
EfficientNetB2	260
EfficientNetB3	300
EfficientNetB4	380
EfficientNetB5	456
EfficientNetB6	528
EfficientNetB7	600

Table 7 list of input shape expected for each model.

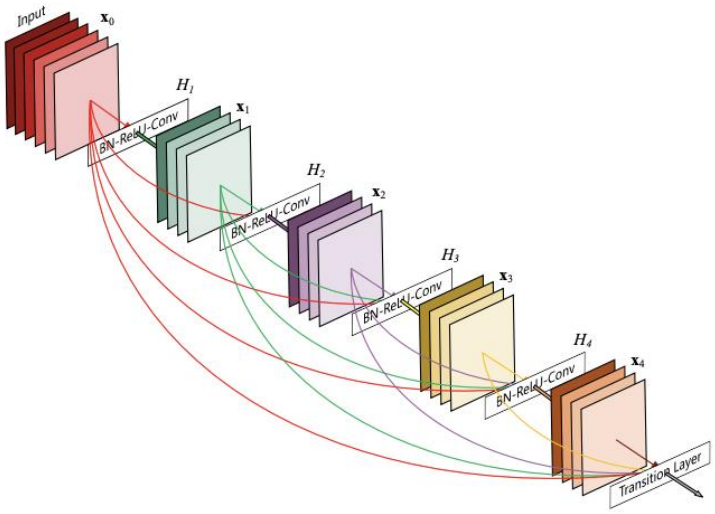


Figure 29: EfficientNetB0 model architecture

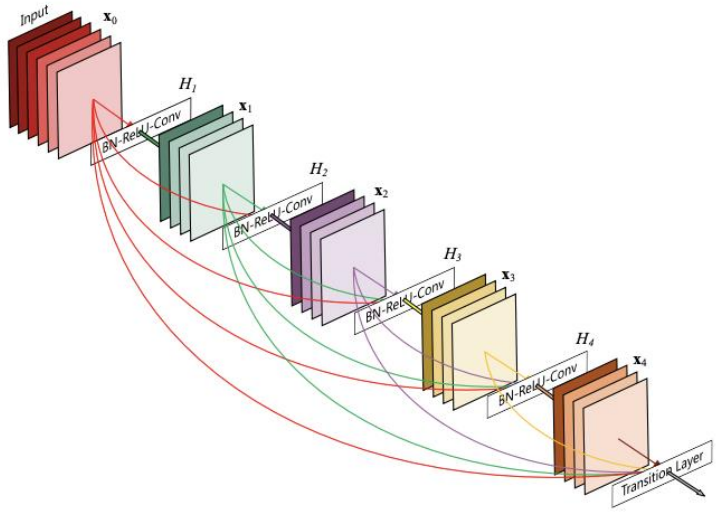


Figure 30: EfficientNetB7 model architecture

4.6 Extracting results

The process of extracting results involves evaluating the performance of a trained machine learning or deep learning model on a test dataset. This process is critical to assess the generalization ability of the model and to identify potential issues or areas for improvement.

There are various techniques that can be applied when extracting results from a machine learning or deep learning model. Some of the common techniques include:

Confusion matrix:

A confusion matrix is a performance measurement tool used in machine learning and statistical classification tasks. It is a table that allows visualization of the performance of a classification model by summarizing the predicted and actual class labels.
In a binary classification problem, the confusion matrix has four components:

1. True Positives (TP): The number of instances that are correctly predicted as positive (belonging to the positive class).
2. False Positives (FP): The number of instances that are incorrectly predicted as positive (predicted as belonging to the positive class, but actually belonging to the negative class).
3. True Negatives (TN): The number of instances that are correctly predicted as negative (belonging to the negative class).
4. False Negatives (FN): The number of instances that are incorrectly predicted as negative (predicted as belonging to the negative class, but actually belonging to the positive class).

Each cell in the matrix represents the count (or proportion) of instances falling into that particular category. By examining the values in the confusion matrix, various performance metrics can be calculated, such as accuracy, precision, recall, and F1 score, which provide insights into the model's performance in terms of correctly identifying positive and negative instances.
The confusion matrix is a valuable tool for evaluating the performance of classification models, particularly in scenarios where imbalanced classes or specific error types need to be analyzed. It helps in understanding the strengths and weaknesses of a model's predictions and can guide further improvements or adjustments in the model or the classification task [28].

Accuracy

Accuracy is used to measure the number of correct predictions in relation to all predictions made for a feature, disregarding whether the prediction was correct or wrong. An accuracy score of 75% to 95% is desirable [29].

The mathematical formula that represented the accuracy can be defined by:

$$Accuracy = \frac{Correct\ prediction}{Total\ cases}$$

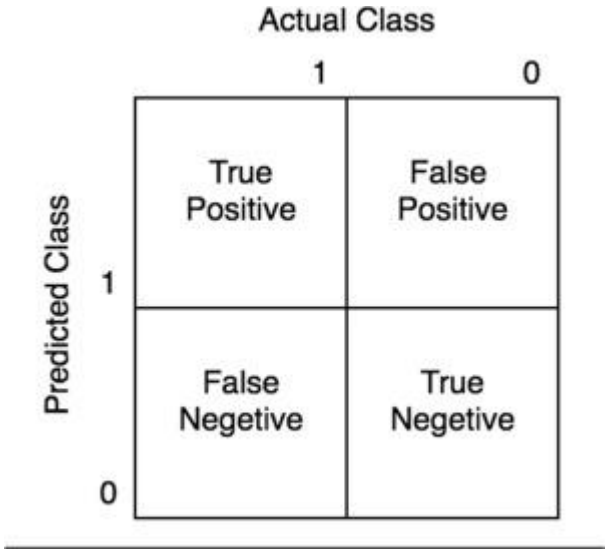


Figure 31: Confusion matrix

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Where:

Patient: positive for the disease

- Healthy: negative for the disease
- True positive (TP) = the number of cases correctly identified as patient
- False-positive (FP) = the number of cases incorrectly identified as patient
- True negative (TN) = the number of cases correctly identified as healthy.
- False-negative (FN) = the number of cases incorrectly identified as healthy.

Loss

It is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater [29].

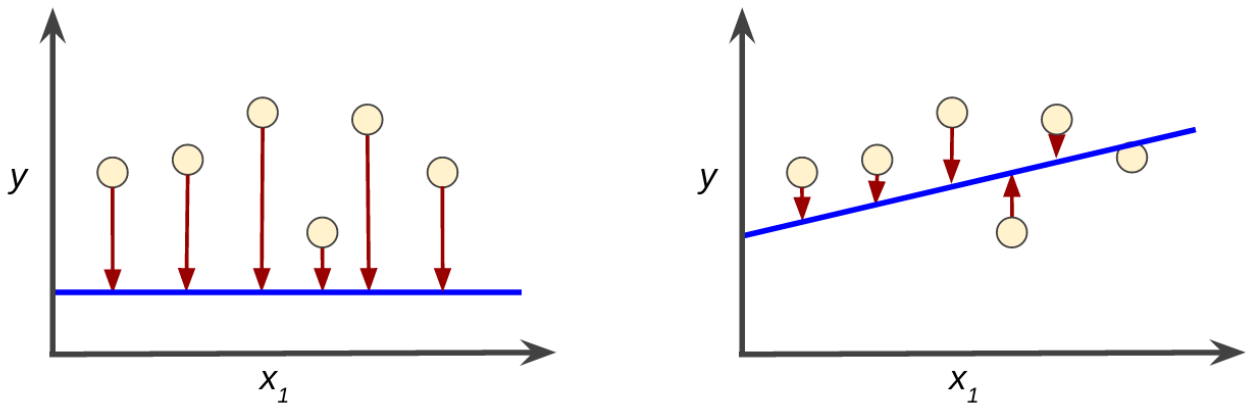


Figure 32: High loss in the left model; low loss in the right model.

Precision

It evaluates how precise a model is in predicting positive labels. Precision answers the question, out of the number of times a model predicted positive, how often was it correct? Precision is the percentage of your results which are relevant [30]. It summarized the exactness' of the model The formula for precision is below:

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Recall

Recall calculates the percentage of actual positives a model correctly identified (True Positive). When the cost of a false negative is high, you should use recall [30]. The formula for recall is below:

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

F-measure

The F-measure displays the relationship between Precision and Recall. It considered the relationship between the number of True/
False Positives and True/ False Negatives [31]. The F1 Score expresses the balance between 'precision' and 'recall

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

4.7 Predicting on new dataset.

When predicting on a new dataset that has not been used to testing, it is important that the data pre-processing is done in exactly the same way as on the training dataset. I therefore initialize all the same pre-processing steps on the image as mentioned earlier, before calling it.

Steps for Predicting classes:

1. Prepare the new dataset: The new dataset should be well-formatted and preprocessed in the same way as the training and testing datasets used to train and evaluate the model.
2. Load the trained model: Load the weights of the trained model into memory using the appropriate function in the deep learning framework.
3. Preprocess the input data: Apply the same preprocessing steps to the input data as were used on the testing data. This ensures that the input data is consistent with the data used during training and testing.
4. Apply the model to the new dataset: Use the predict function of the model to obtain the predicted labels or values for each input sample in the new dataset.
5. Interpret the predictions: Examine the predictions of the model and interpret them in the context of the specific task.
6. Evaluate the model performance (optional): If the new dataset contains true labels, the performance of the model on this dataset can be evaluated using the same metrics used to evaluate the model during training, such as accuracy, precision, recall, and F1-score.

4.8 Saving weights of used model.

To save the trained weights of a deep learning model for future use, we store the weights in the .h5 format using the **save** method of the model object in various deep learning frameworks, such as Keras. Once the trained weights are saved in the .h5 format, they can be later loaded into the model using the **load_weights** method of the model object.

Additionally, during the training process, we establish checkpoints to preserve the best performing model. This can be accomplished by utilizing the **ModelCheckpoint callback** in Keras, which enables us to save the model weights at regular intervals during training and choose the model with the highest validation metrics, such as the lowest validation loss. By including the **checkpoint** object as a callback in the **fit** method of the model, we can save the best performing model in the .h5 format for future use.
Overall, saving the trained weights in the .h5 format and implementing checkpoints for preserving the best model during training provide benefits. This allows the model to be reused for subsequent predictions without the need to retrain it from scratch and enables the selection of the best performing model based on validation metrics during training.

implementation steps:

1. Train your model: Train deep learning model using training data.
2. Choose a file format: Decide on the file format in which you want to save the model weights using the .h5 format.
3. Save weights.

4.9 Real time prediction

Real-time prediction using a camera involves using the weights of a pre-trained deep learning model to make predictions on live video from a camera.

In real-time prediction using a camera, we capture live video from the camera and preprocess each frame of the video in the same way as the training and testing data. We then apply the pre-trained model to each frame and obtain the predicted labels or values for the input frame. This process is repeated for each new frame captured by the camera, allowing us to make real-time predictions on live video.

Real-time prediction using a camera can be implemented using a deep learning framework TensorFlow or PyTorch, along with a camera library OpenCV.

Steps to Perform real time Facial Expression Recognition

Step-1: Detect the faces in the input video stream.
Step-2: Find the Region of Interest (ROI) of the faces.
Step-3: Apply the Facial Expression Recognition model to predict the expression of the person.
We are using seven Classes here that is 'Angry', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise' ,'Disgust'. So, the predicted images will be among these classes.

4.10 mobile application

4.10.1 Developing Main activity.

It deals with the UI and the user interactions to the screen. In other words, it is a User Interface that contains activities. These can be one or more depending upon the App. It starts when the application is launched. At least one activity is always present which is known as MainActivity [33].

An activity facilitates the following key interactions between system and app:

- Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.
- Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.
- Helping the app handle having its process killed so the user can return to activities with their previous state restored.
- Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows. (The most classic example here being share [33] [34].

Activity lifecycle:

onCreate: This is the first callback and called when the activity is first created.

onStart: This callback is called when the activity becomes visible to the user. It is followed by onResume() if the activity is invoked from the background. It is also invoked after onCreate() when the activity is first started.

onResume: This is called when the user starts interacting with the application. At this point, the activity is at the top of the activity stack, with a user interacting with it.

Always followed by onPause() when the activity goes into the background or is closed by the user.

onPause: The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.

onStop: This callback is called when the activity is no longer visible.

onDestroy: This callback is called before the activity is destroyed by the system.

onRestart: This callback is called when the activity restarts after stopping it [35].

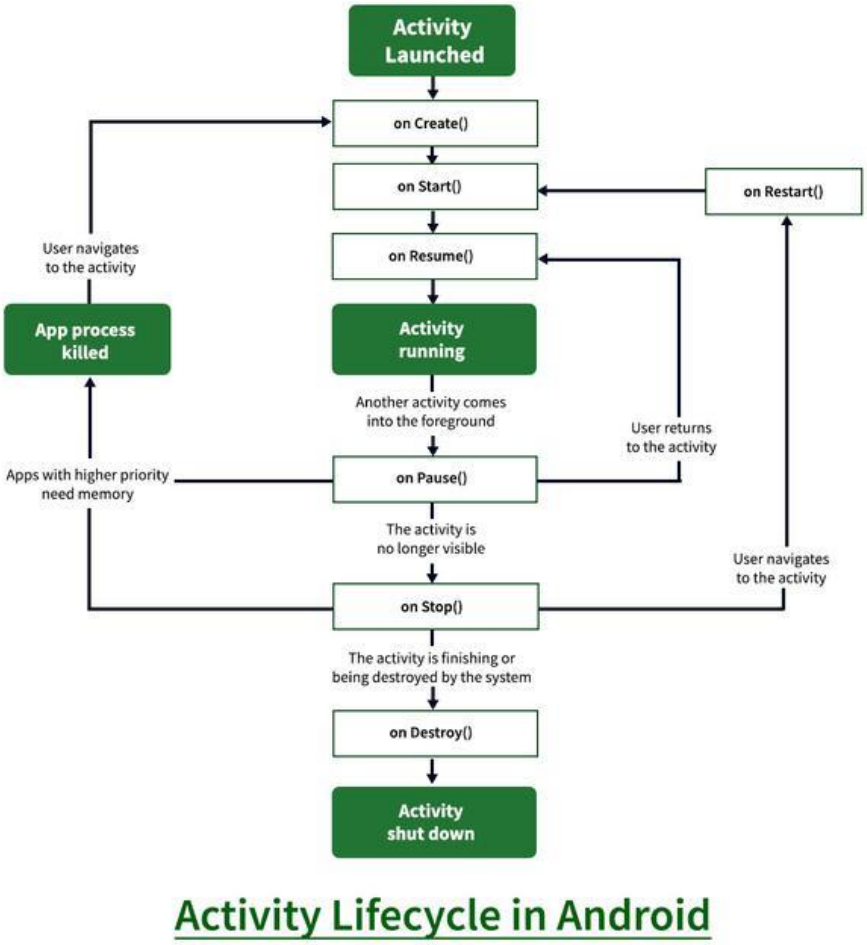


Figure 33: ***Activity lifecycle***

- We have built the app with the main activity set to open in the home fragment. This approach provides a seamless user experience, allowing users to easily access the app's main features and functionality from the home screen. By prioritizing the

home fragment as the main activity, we aim to enhance the app's usability and accessibility for our users.

4.10.2 Android UI Layouts

Android **Layout** is used to define the user interface that holds the UI controls or widgets that will appear on the screen of an android application or activity screen.
Types of Android Layout:

Android Linear Layout: LinearLayout is a ViewGroup subclass, used to provide child View elements one by one either in a particular direction either horizontally or vertically based on the orientation property.
Android Relative Layout: RelativeLayout is a ViewGroup subclass, used to specify the position of child View elements relative to each other like (A to the right of B) or relative to the parent (fix to the top of the parent).

Android Constraint Layout: ConstraintLayout is a ViewGroup subclass, used to specify the position of layout constraints for every child View relative to other views present. A ConstraintLayout is similar to a RelativeLayout but having more power [36].

Linear Layout

Linear layout is a simple layout used in android for layout designing. In the Linear layout all the elements are displayed in linear fashion means all the childs/elements of a linear layout are displayed according to its orientation. The value for orientation property can be either horizontal or vertical.

There are two types of linear layout orientation:
1. Vertical
In this all the children are arranged vertically in a line one after the other.
2. Horizontal
In this all the children are arranged horizontally in a line one after the other [36].

4.10.3 Android UI component

Android UI Controls are those components of Android that are used to design the UI in a more interactive way. It helps us to develop an application that makes user interaction better with the view components. Android provides us with a huge range of UI controls of many types such as buttons, text views, etc.

TextView: Used for displaying text content to the user.

EditText: Allows the user to input text or numbers.

Button: Triggers an action when tapped by the user.

CheckBox: Presents a binary choice option that can be selected or deselected.

RadioButton: Offers a set of mutually exclusive options, where only one option can be selected at a time.

ToggleButton: Represents a two-state button that toggles between on and off states.

ProgressBar: Displays a visual indication of the progress of an ongoing task.

ImageView: Used for displaying images or icons.

DatePicker: Allows the user to select a specific date.

TimePicker: Enables the user to choose a specific time.

SeekBar: Provides a horizontal bar that the user can slide to select a value within a specified range.

ImageView: used to display images within an app. It allows developers to show different types of images, such as photos, icons, or graphics [37].

Spinner: used to select an item from a list of items. When the user taps on a spinner a drop-down menu is visible to the user [38].

Figure 34: Android UI component

Our Android UI Layouts

In our Android application, we utilize various Android UI layouts and components to create an intuitive and visually appealing user interface. Here are the key layouts and components we employ:

Relative Layout:
Used to specify the position of child View elements relative to each other like.

TextView:
to display text content to the user. It allows us to present information, labels, instructions, or any textual data within the application interface.

ImageView:
to display images within our application

Spinner:
used to create a dropdown selection menu.

Bottom Navigation Bar:
The Bottom Navigation Bar is an essential UI component for navigation purposes. It typically appears at the bottom of the screen and offers a set of icons or labels representing different app sections or destinations. It allows users to switch between app sections easily, providing a consistent and intuitive navigation experience.

4.10.4 Developing Fragment

It represents the UI portion of application screen. A fragment has its own layout, its own lifecycle, can also handle the input from UI and represent the outcome.
Fragment can't live / create on their own- They must be hosted by an activity.

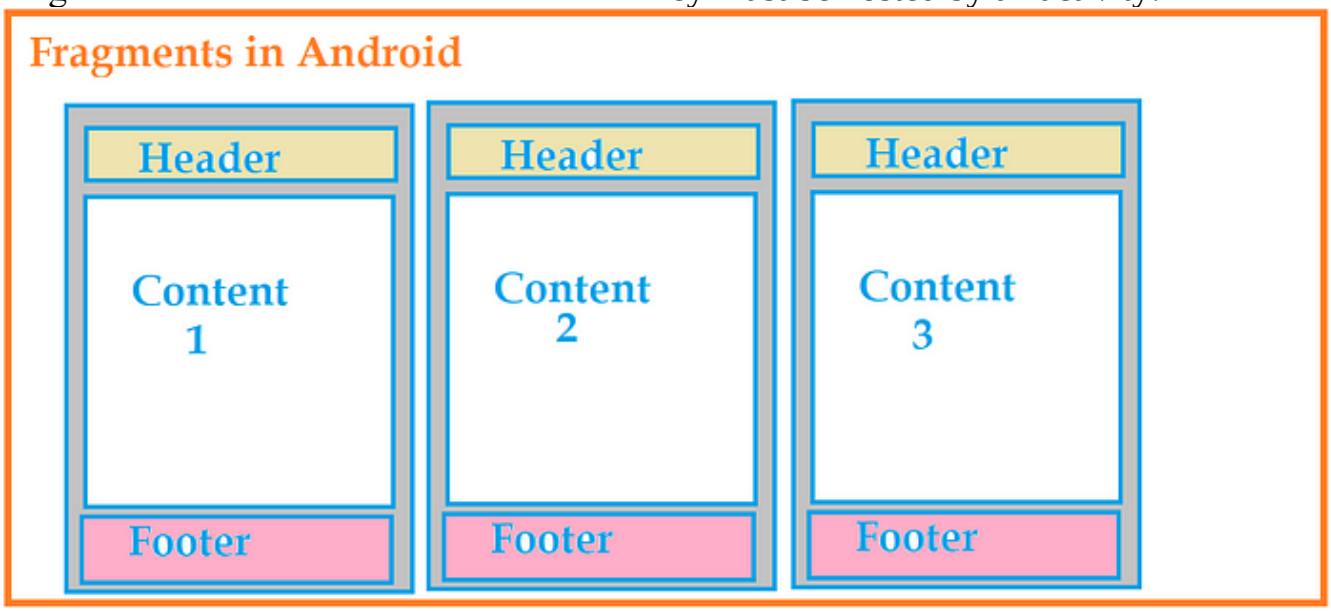


Figure 35: Fragment

lifecycle of Fragment:

- onCreate() : here fragment got created.
- onCreateView(): here fragment views got created.
- onActivityCreated(): this will help to initiate the view but now it got deprecated. Alternative of this lifecycle method is onViewCreated()
- onStart(): it invokes when fragment got visible.
- onResume(): this method invokes when fragments start interacting.
- onPause(): this method invokes when fragment stop interacting.
- onStop() : this method invokes when fragments are no longer visible.
- onDestroyView(): this method destroys the view of it.
- onDestroy(): this method invokes to destroy fragment.
- onDetach(): this method invokes detaching reference with hosted activity [39].

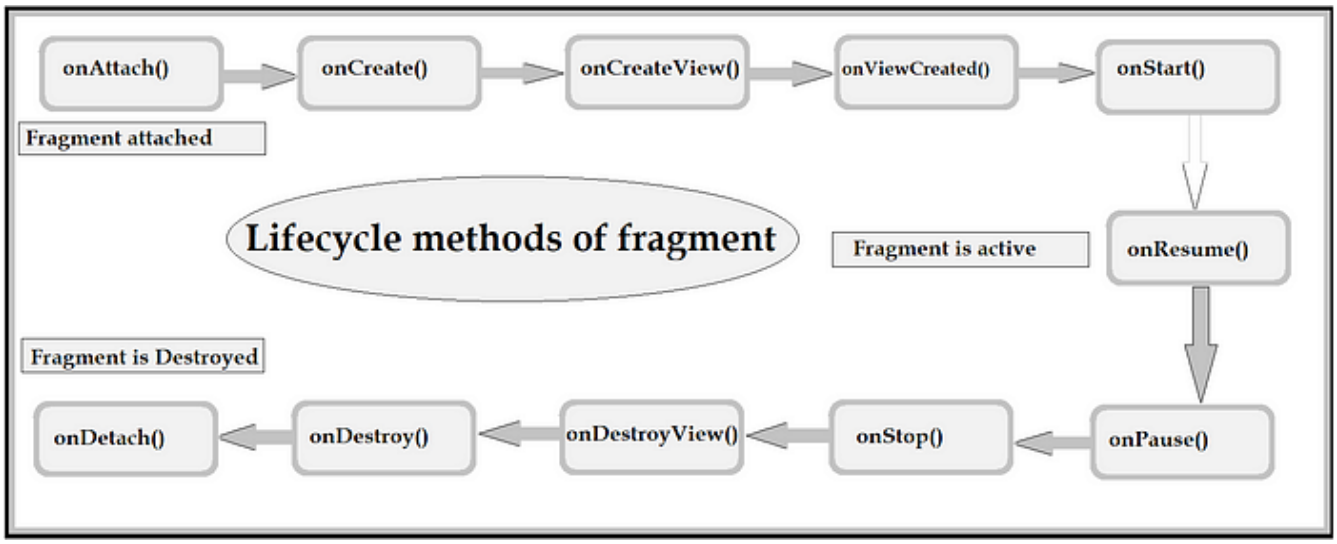


Figure 36: ***lifecycle of Fragment***

Types of Fragments:

- **Single frame fragments**: Single frame fragments are used for hand hold devices like mobiles, here we can show only one fragment as a view.
- **List fragments**: fragments having special list view are called as list fragment.
- **Fragments transaction**: Using with fragment transaction. we can move one fragment to another fragment.

Fragment Container View:

Fragment Container View is a specialized custom View that extends FrameLayout and is designed specifically for hosting fragment views. It differs from other ViewGroups in that it only accepts fragment views and provides enhanced flexibility for fragment transaction [40].it is a subclass of Frame Layout.

fragment transactions

- **Add:** (containerViewId: Int, fragment: Fragment, tag: String): adds a fragment not previously added, associated with given tag, to the given container.
- **Remove: (**fragment: Fragment): removes a fragment and its associated state from the fragment manager.
- **Replace:** (containerViewId: Int, fragment: Fragment, tag: String): Removes the current fragment and replaces it with the new one provided.
- **Hide:** (fragment: Fragment): Hides an existing fragment. This is only relevant for fragments whose views have been added to a container, as this will cause the view to be hidden.

- **Show:** (fragment: Fragment): Shows a previously hidden fragment. This is only relevant for fragments whose views have been added to a container, as this will cause the view to be shown [41].

4.10.5 Bottom Navigation Bar Android:

Bottom navigation bars make it easy to explore and switch between top-level views in a single tap [42].

- In the development of our application, we have implemented three distinct fragments to cater to the functionalities of the home, settings, and education screens. These fragments serve as modular components that encapsulate the user interface and logic for each respective screen. Additionally, we have integrated a BottomNavigationView to facilitate navigation between these fragments.

To enhance the user experience, we have ensured that the BottomNavigationView consistently displays both icons and text for all three items. By leveraging the capabilities of BottomNavigationView, we have successfully created a seamless and intuitive navigation system.

This design decision ensures that users can easily identify and select the desired screen by associating the corresponding icon and text. Furthermore, by providing a clear visual representation of each screen's purpose, we aim to improve usability and engagement within our application.

By adhering to these practices, we have established a professional and user-centric approach to fragment-based development, enhancing the overall functionality and appeal of our application.

Implementation steps

1. Creating the fragment classes and designing their respective layout XML files.
2. Setting up the BottomNavigationView in the activity layout, adding menu items for each fragment with corresponding icons and text.
3. Handling fragment transactions using a FragmentManager and FragmentTransaction in response to BottomNavigationView item selections.
4. Implementing fragment state saving by overriding the onSaveInstanceState() method and restoring the saved state in the fragment's lifecycle methods.
5. Customizing the icon and text display in the BottomNavigationView by setting itemIconTintList and itemTextColor attributes using a selector XML file.

4.10.6 Settings fragment

In order to enhance the user experience within the app, a systematic approach was adopted by providing users with the ability to customize their preferences within the settings fragment,particularly in choosing

1. Type

(Male or Female)

2.Image Type

(Cartoon, real, or Emojis).
This approach ensures that users have control over their personal experience within the app, resulting in greater personalization and increased user satisfaction."

RFCOMM:

RFCOMM (Radio Frequency Communication) is a connection-oriented, streaming transport protocol used over Bluetooth. It is the most common type of Bluetooth socket and is supported by the Android APIs. RFCOMM is also known as the Serial Port Profile (SPP) and provides a reliable data connection between Bluetooth-enabled devices [43].

As a connection-oriented protocol, RFCOMM establishes a virtual serial port communication channel over Bluetooth. It allows for the transmission of data streams between devices, similar to how data is sent over a physical serial port connection. RFCOMM provides a convenient way to establish a reliable and standardized communication channel between Bluetooth devices.

In Android, RFCOMM is implemented using the BluetoothSocket class. This class allows applications to establish an RFCOMM connection, send and receive data, and manage communication between Bluetooth devices.

RFCOMM is widely used for various applications, such as wireless file transfer, Bluetooth-based audio streaming, and device-to-device communication. Its support for the Serial Port Profile makes it compatible with a wide range of Bluetooth devices and enables seamless communication between them [44].

3. **Connect Button**

- allows users to establish connections with other devices or contacts using various communication methods, such as Bluetooth or Wi-Fi Direct.
- When the user taps the contact button, it can trigger a dialog or screen where they can search for nearby devices or contacts to connect with.
- Once a connection is established, users can initiate communication, share data, or perform specific actions based on the functionality provided by the application.

4. **Server Button**

- enables users to set up their device as a server or host, allowing other devices or clients to connect to it.
- When the user selects the server button, it can present options for setting up network connections, such as Wi-Fi hotspot functionality or hosting services using specific protocols (e.g., HTTP, FTP).
- Once the server is activated, other devices can connect to it and access resources or services provided by the application running on the server device.

By incorporating the contact and server buttons in the settings fragment, we can establish connections with external devices, emotion detection model through connection within a laptop via Bluetooth. The contact button allows us to search and connect with nearby devices, while the server button enables the device to act as a host for external systems. This integration facilitates seamless communication, data exchange, and real-time updates between the application and the external devices. Overall, it enhances the application's functionality.

4.10.7 Home fragment

In our project, in the home fragment, after receiving the output of the emotion detection model, the corresponding images reflecting the detected emotion are displayed. This allows users to visually perceive and understand the emotions captured by the model.

Additionally, the project includes a feature where the settings made by the user are transmitted to the home fragment. These settings are then applied in real-time, modifying the behavior or appearance of the home fragment based on the user's preferences. This ensures a personalized and tailored experience for each user.

By integrating the emotion detection output and the transmission of settings to the home fragment, our project aims to provide a dynamic and interactive user interface that effectively represents emotions and responds to user customization.

- So, we utilized ImageView, which is a UI component used to display images within an app. It provides developers with the ability to showcase various types of images, including photos, icons, or graphics.

Implementation steps

1. Add the ImageView element to XML layout file, specifying its position, size, and other desired attributes.
2.Assign an image resource to the ImageView by uploading specific images for each type, such as male or female, and image types like cartoon, real, or emoji. This can be done programmatically in your code or by specifying the image resource directly in the XML layout file using the src attribute.
3.Customize the appearance and behavior of the ImageView as needed.

4.10.8 Education fragment

It is designed for educational purposes within the app. Although currently empty, it serves as a placeholder for future development and will be established in subsequent work.

Android Emulator

We utilized the Android Emulator to establish and test our application .it is a powerful tool that allows developers to simulate Android devices on their computers for testing and debugging applications. With the emulator, we test apps on a wide range of virtual devices and Android API levels without the need for physical devices. It offers several advantages that make it an essential component of the Android development process.

The Android Emulator is a versatile tool that caters to the diverse testing needs of Android app developers. Whether you are developing for phones, tablets, wearables, or other Android-based devices, the emulator offers a reliable and convenient platform to validate and refine your app's performance [45].

4.11 Hardware connection

4.11.1 Camera module

ESP32-CAM WiFi – WiFi Module ESP32 serial to WiFi ESP32 CAMERA Development Board – 5V Bluetooth with OV2640 Camera Module [50].

Features

- Onboard ESP32-S module, supports WiFi + Bluetooth + OV2640 Camera
- Low-power dual-core 32-bit CPU for application processors
- Main frequency up to 240MHz, computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4M PSRAM
- Supports interfaces such as UART/SPI/I2C/PWM/ADC/DAC
- Include Camera OV2640 and can support OV7670 camera (not included) , built-in flash
- Support image WiFi upload
- Support TF card
- Support multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Support STA/AP/STA+AP working mode.
- Support Smart Config/AirKiss one-click distribution network.
- Support secondary development.

Figure 37: Camera module

4.11.2 Arduino uno

Arduino Uno was utilized as the hardware platform to upload the code from the IDE to the ESP module.

Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst-case scenario you can replace the chip for a few dollars and start over again [52].

Figure 38: Arduino

4.11.3 Powering of Esp32 Camera

Using 2 lithium-ion batteries with their holder and type c charger module for lithium-ion batteries charging, then casing all components by making a 3d printing box with a specific dimension.

Figure 39: powering components

4.11.4 Total Cost

ESP camera module	450
Arduino Uno	365
Lithium batteries	100
Type C Module	35
Battery holder	10
Casing	120
Jumpers	15
Total	1,095

Table 8: cost

4.11.5 Connection between camera and model

The connection between the ESP32-CAM WiFi module, which includes the ESP32 microcontroller and the OV2640 camera module, and the laptop for real-time image transmission to a facial expression recognition model was established through a WiFi network.

First, the ESP32-CAM WiFi module connected to the local WiFi network using the appropriate credentials. Once connected, it transmitted the captured images in real-time to the laptop over the WiFi connection.

On the laptop side, a facial expression recognition model was established. MobileNetV2 model weights were used as the deep learning model for facial expression recognition. This model was trained to analyze and detect facial expressions in real-time.

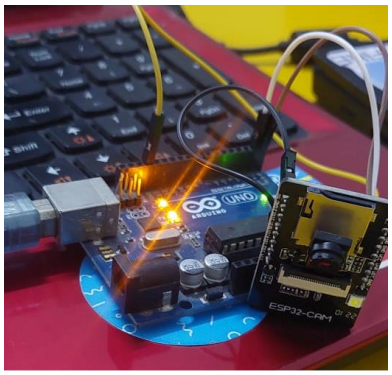


Figure 40 the connection of Arduino with esp32 camera

To enable real-time image transmission from the ESP32-CAM to the laptop, the ESP32-CAM captured images using the OV2640 camera module. The captured images are then saved to the laptop from the stream of the camera. This stream is accessible through the local IP of the module.

On the laptop, images are received from the ESP32-CAM and fed into the MobileNetV2 facial expression recognition model. The model analyzes the received images and detects facial expressions in real-time.

The WiFi connection between the ESP32-CAM and the laptop allowed for seamless and wireless transmission of the images. It enabled real-time monitoring and analysis of facial expressions using the MobileNetV2 facial expression recognition model running on the laptop.

This setup provided a convenient and efficient way to capture images using the ESP32-CAM WiFi module and process them in real-time on the laptop using the MobileNetV2 facial expression recognition model. The WiFi connection ensured a reliable and fast data transmission between the ESP32-CAM and the laptop.

4.11.6 Connection between model and application

The connection between the facial expression recognition model, which is established on the laptop, and the application is established using the previously explained RFCOMM module. RFCOMM is a Bluetooth protocol that provides a connection-oriented, streaming transport over Bluetooth.

In this setup, the camera captures images, as explained earlier. The facial expression recognition model, running on the laptop, receives these images and processes them to detect emotions.

Once the emotions are detected, the results are sent back to the application over the established Bluetooth connection using the RFCOMM module. The application, based on the user's preferences and settings (such as male or female, cartoon or real), then displays the results accordingly.

The RFCOMM module ensures a reliable and efficient transmission of data between the laptop and the application. It allows for real-time communication and synchronization of data, enabling the application to receive and display the emotion detection results from the laptop in accordance with the user's preferences.

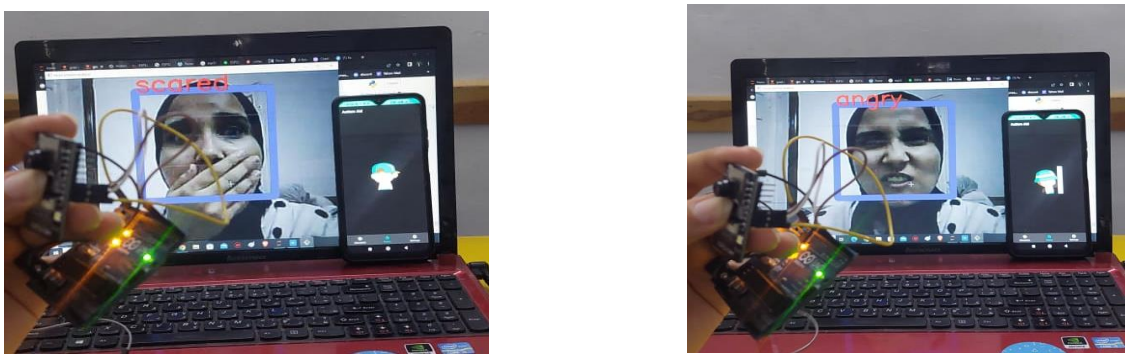


Figure 41: Esp32 camera and mobile application communication

Figure 42: the glasses

Chapter 5

Results & Discussion

5.1 Dataset

We used The FER2013 dataset. It includes 35,887 grayscale images of 48x48 pixels resolution, representing seven different facial expressions: anger, disgust, fear, happiness, sadness, surprise, and neutral. The dataset is divided into a training set, a public test set, and a private test set, with a total of 28,709 images for training and 7,178 images for testing.

Each image is labeled with the corresponding emotion category, enabling supervised learning approaches for training facial expression recognition models. The dataset's size, diversity, and standardized labeling make it a valuable resource for advancing the field of emotion recognition from facial expressions.

5.2 Pre-processing phase

The preprocessing steps employed in the proposed method include under sampling and data augmentation to address class imbalance in the dataset.

First, under sampling is performed to decrease the proportion of the majority class until its number of samples is similar to that of the minority class (disgust). This helps to balance the class distribution and prevent bias towards the majority class.

Next, data augmentation techniques are applied to further balance the dataset. All classes are up sampled to a size of 2500 using techniques such as intensity variation and flipping. This involves creating new samples by applying these transformations to the existing images, ensuring consistency and fairness across all classes.

The dataset is then divided into three sets: a training set, a testing set, and a validation set, using an 80-10-10 split. This division allows for effective evaluation of the machine learning model's performance and helps prevent overfitting.

In the data processing stage, the images are converted to numpy arrays to facilitate further processing and analysis. Additionally, normalization, a common data preparation technique in machine learning, is applied to scale the pixel values of the images to a consistent range.
Overall, the proposed preprocessing method aims to address class imbalance, enhance dataset balance through undersampling and data augmentation, ensure fair representation of all classes, and prepare the data for training and evaluation through splitting, image-to-array conversion, and normalization.

5.3 Model phase

In this project, we explored different convolutional neural network (CNN) models, including VGG16, ResNet50, DenseNet201, MobileNet, EfficientNetB0, and EfficientNetB7, to improve the performance of facial emotion recognition. We started by implementing a baseline CNN model with a specific structure, as described previously.

Next, we investigated the performance of pretrained models by utilizing transfer learning. Transfer learning allows us to leverage the learned features from models that were trained on large datasets, such as ImageNet, and apply them to our specific task of facial emotion recognition. This approach can save training time and potentially improve the accuracy of our models.

Among the pretrained models we evaluated, VGG16, ResNet50, DenseNet201, MobileNet, EfficientNetB0, and EfficientNetB7, we observed an increase in performance compared to the baseline model. These models have been widely used and proven effective in various computer vision tasks.

Through the evaluation of these pretrained models, we gained insights into their strengths and weaknesses for facial emotion recognition. The choice of the most suitable model depends on factors such as the available computational resources, real-time requirements, and desired accuracy.

5.3.1 CNN Model

1. CNN model (Before applying processing)

Below is the plotted training history for model accuracy and model loss using the VGG16 model. The model was trained for 80 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

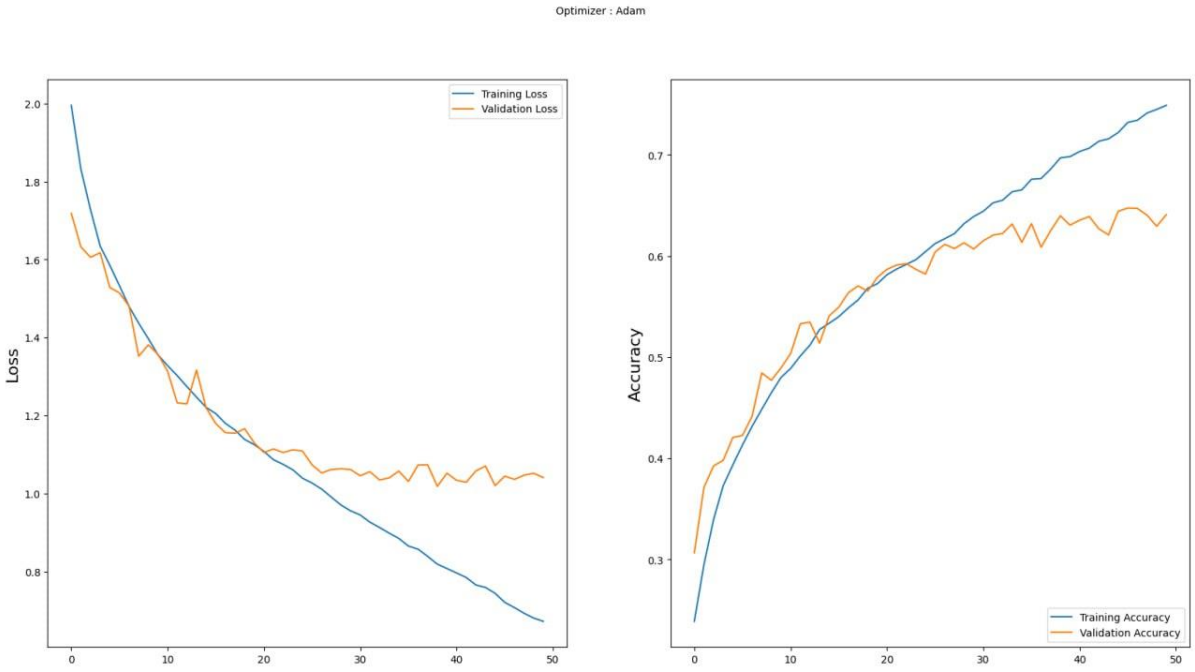


Figure 43: accuracy and loss history for CNN model (Before applying processing)

Confusion matrix:

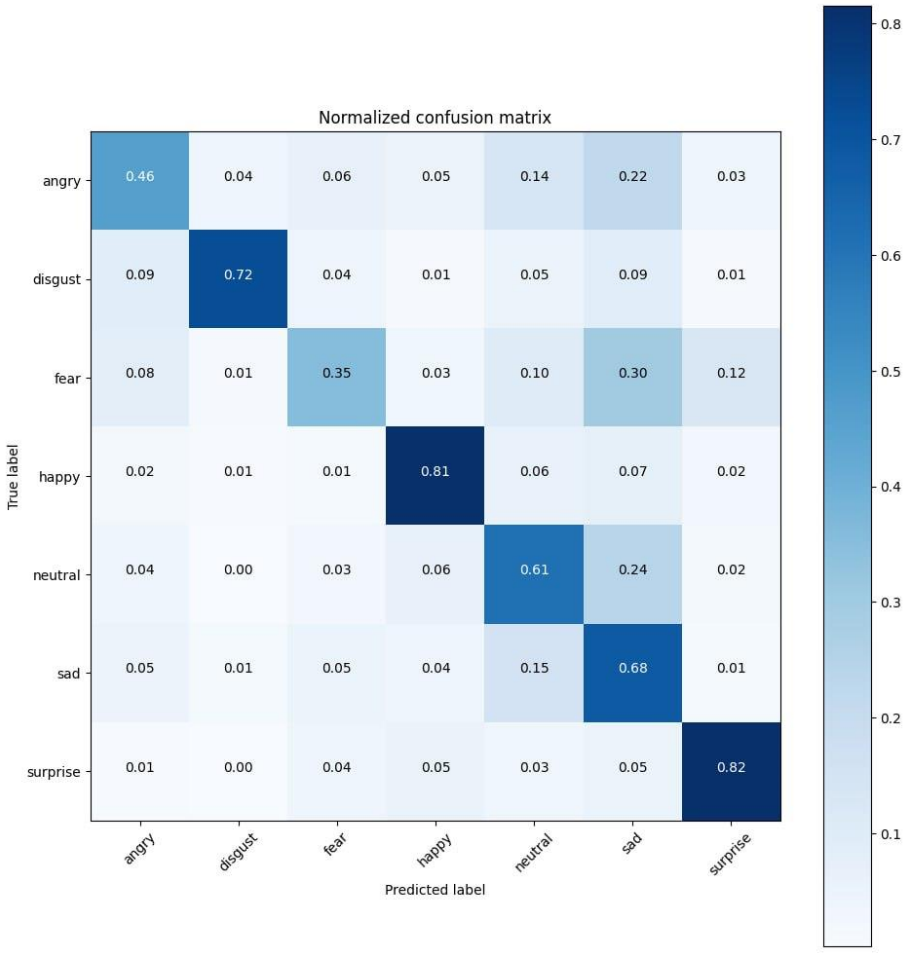
A Confusion Matrix will show a breakdown of all predictions made for an unseen dataset and is very useful for presenting the accuracy of a model with several feature classes.

The table below presents cells for predictions on the horizontal x-axis and cells for groups of correct answers on the vertical y-axis. Contents of cells are fractions of predictions for each emotion. If all predictions were correct, all numbers on the matrix diagonal would be 1.0.

The confusion matrix obtained from the CNN model (before applying processing), trained for 80 epochs, highlights the highest accuracy of 82% in predicting "surprise" labels and the "happy" class. However, the "fear" class had the lowest accuracy of 35%.

Additionally, the overall accuracy of the model was 64%, indicating a relatively lower level of performance. This suggests that the model achieved accurate classifications for the majority of the emotion classes, but there is still room for improvement in accurately predicting all classes.

Figure 44: confusion matrix for CNN model during 80 epochs before preprocessing.



Classification report:

From Scikit-learn, a standardized report for working with classification problems is easily generated. It gives a quick overview of the model accuracy by using standardized measures. The classification report function display numerical representations of F measures such as precision, recall, f1-score, and support for each feature class.

label	precision	recall	F-score
Angry	0.54	0.54	0.54
Disgust	0.49	0.66	0.56
Fear	0.55	0.43	0.48
Happy	0.83	0.84	0.83
Neutral	0.61	0.53	0.56
Sad	0.47	0.57	0.52
Surprise	0.72	0.80	0.76

Table 9: Classification report from using CNN model on 80 epochs before processing.

The classification report for CNN before processing reveals the metric results obtained by comparing the actual classes to the predicted classes. The "Happy" class achieved the highest scores, approximately 83%, for precision, recall, and F1 score. In contrast, the "sad" emotion received the lowest score 47%.

5.2 CNN model (After applying processing)

Below is the plotted training history for model accuracy and model loss using the VGG16 model. The model was trained for 50 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

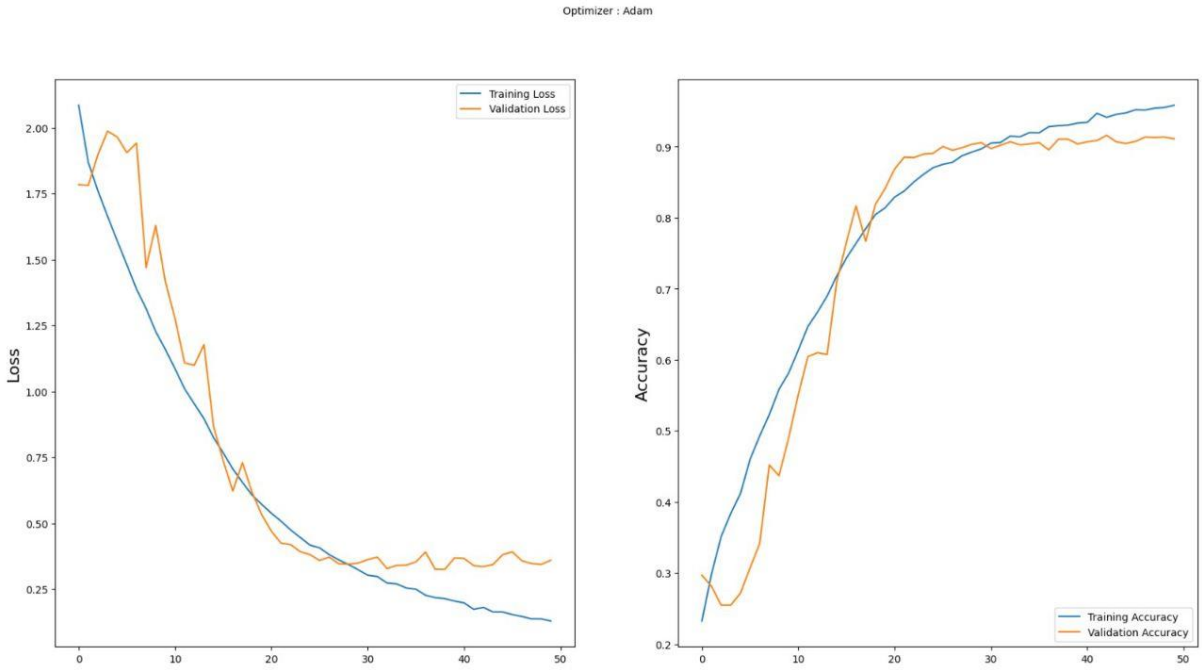


Figure 45: accuracy and loss history for CNN model (After applying processing)

Confusion matrix

The confusion matrix obtained reveals the highest accuracy of 94% in predicting "disgust" and "surprise" labels for the CNN model, which was trained for 50 epochs. Additionally, the accuracy for the other classes is approximately 90%.

Furthermore, the overall accuracy of the model was 90.40%. This signifies a high level of performance, indicating that the model achieved accurate classifications for the majority of the emotion classes.

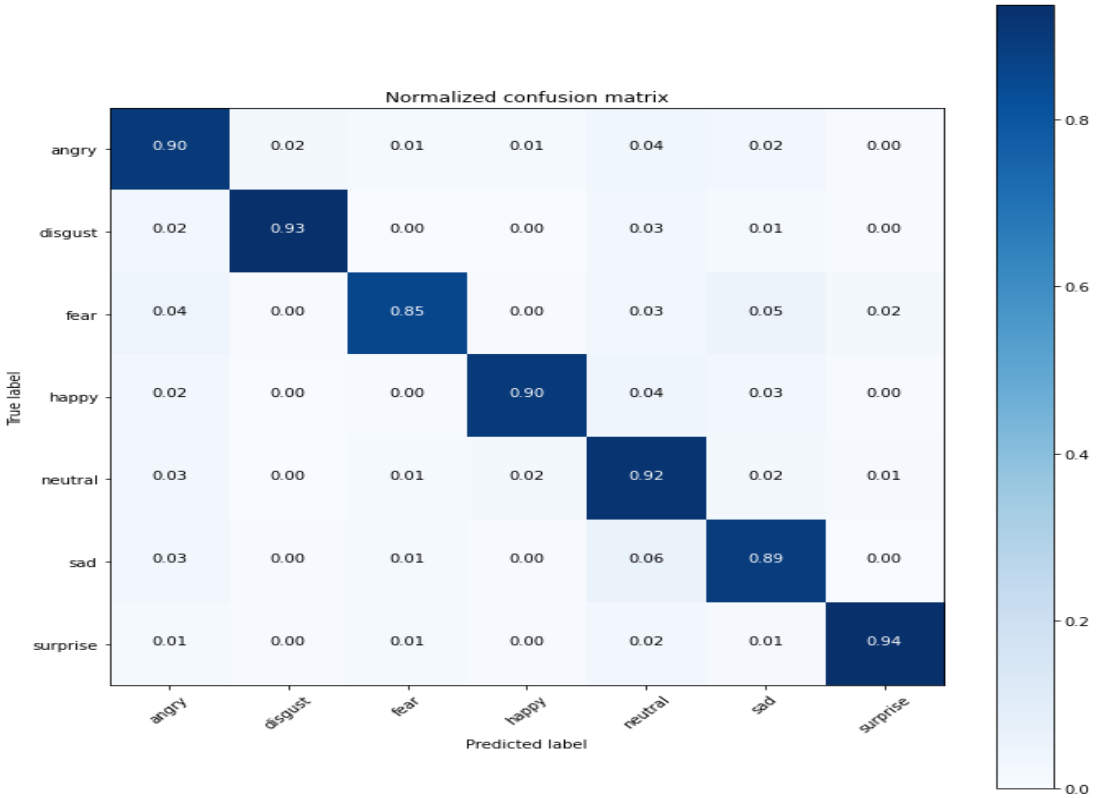


Figure 46: confusion matrix for CNN model during 50 epoch after preprocessing.

Classification report

The classification report function shows numerical F-measures such as precision, recall, f1-score, and support for each class.

label	precision	recall	F-score
Angry	0.848	0.896	0.871
Disgust	0.970	0.932	0.951
Fear	0.942	0.852	0.895
Happy	0.957	0.900	0.928
Neutral	0.809	0.920	0.861
Sad	0.864	0.892	0.878
Surprise	0.962	0.936	0.95

Table 10: Classification report from using CNN model on 50 epochs after processing.

The classification report for the CNN model after processing reveals the metric results obtained by comparing the actual classes to the predicted classes. The "Disgust" class achieved the highest scores, approximately 97%, for precision, recall, and F1 score. The other classes achieved an average score of 90%, indicating a strong overall performance for the model.

5.3.2 Pretrained models

After applying transfer learning and utilizing a lower number of epochs, we observed a notable improvement in the performance of the model. We compared this approach with a conventional CNN model to assess their respective performances.

1. VGG16 model

Metric results

Below is the plotted training history for model accuracy and model loss using the VGG16 model. The model was trained for 50 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

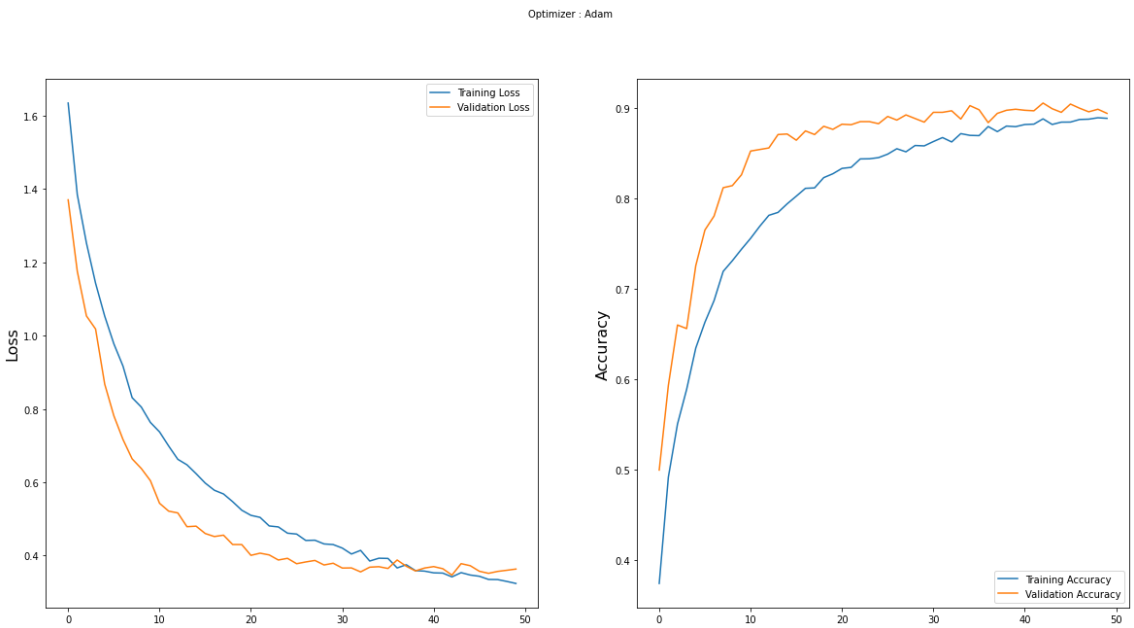


Figure 47: accuracy and loss history for VGG16 model

Confusion matrix

The confusion matrix obtained reveals the highest accuracy of 94% in predicting "disgust" labels for the VGG16 model, which was trained for 50 epochs. Additionally, the accuracy for the other classes is approximately 85%

Furthermore, the overall accuracy of the model was **87.54**%. This signifies a high level of performance, indicating that the model achieved accurate classifications for the majority of the emotion classes.

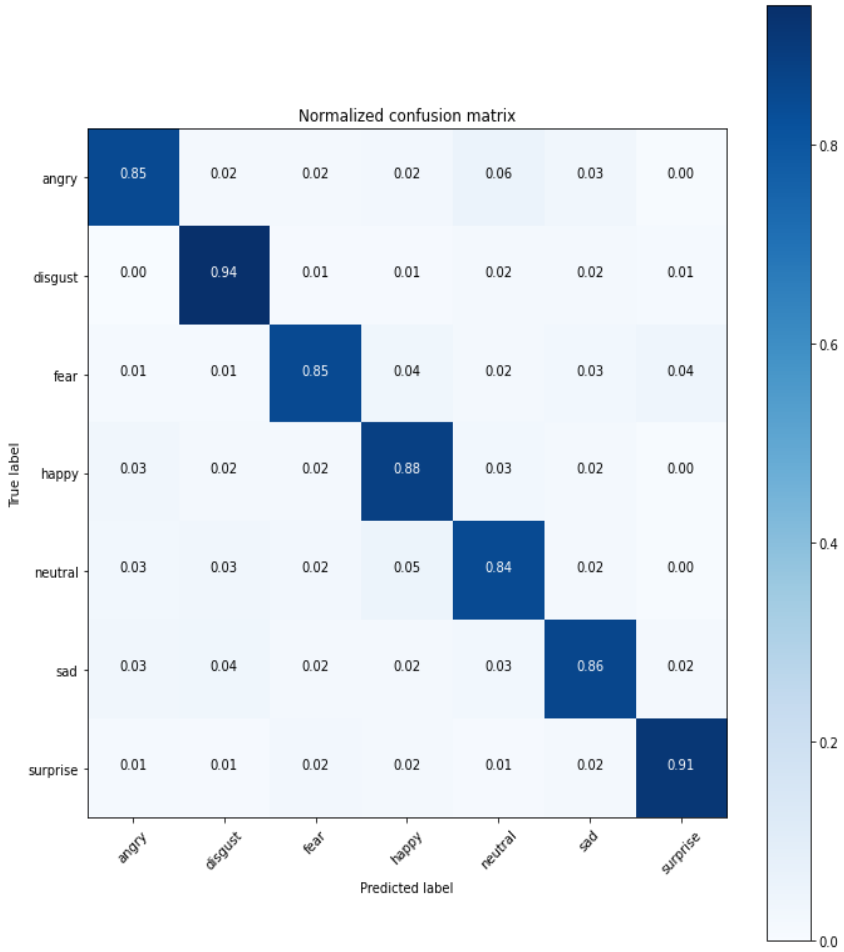


Figure 48: confusion matrix for VGG16 pre_trained model during 50 epochs

Classification report

The classification report function shows numerical F-measures such as precision, recall, f1-score, and support for each class.

label	precision	recall	F-score
Angry	0.879	0.848	0.863
Disgust	0.880	0.94	0.909
Fear	0.890	0.848	0.868
Happy	0.856	0.88	0.867
Neutral	0.844	0.844	0.844
Sad	0.863	0.856	0.859
Surprise	0.915	0.912	0.913

Table 11: Classification report from using VGG16 model.

The classification report for VGG16 shows the metric results comparing the actual classes to the predicted classes. The "surprise" class achieved the highest scores of around 91% for precision, recall, and F1 score. On the other hand, the "natural" emotion received the lowest score of 84.4% for all metrics.

2. ResNet50 model

Metric results

Below is the plotted training history for model accuracy and model loss using the ResNet50 model. The model was trained for 50 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

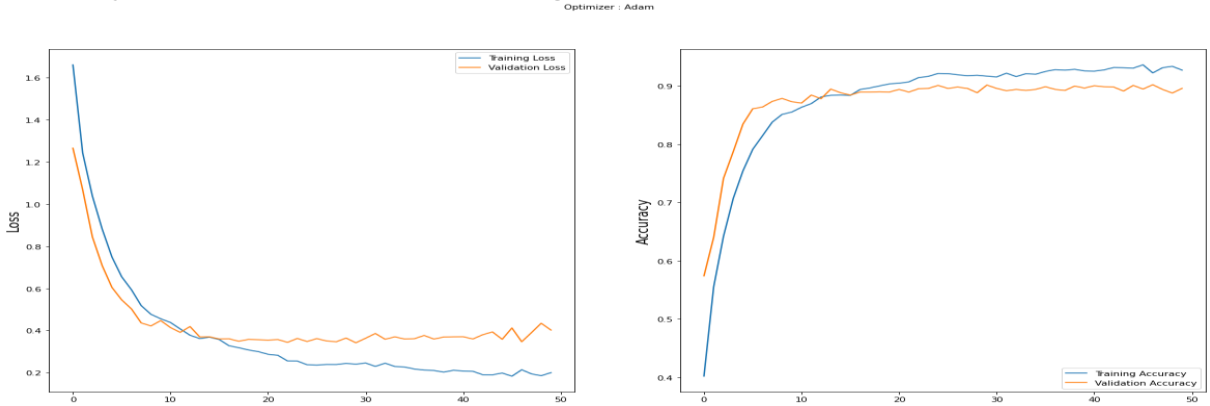


Figure 399: accuracy and loss history for ResNet50 model

Confusion matrix

The confusion matrix obtained from the ResNet50 model, trained for 50 epochs, revealed a highest accuracy of 92% in predicting the 'surprise' and 'happy' labels. Furthermore, the model achieved an accuracy of around 88% for the other classes.

The model achieved an impressive overall accuracy of **88**%, underscoring its exceptional performance. This implies that the model accurately classified a majority of the emotions under consideration.

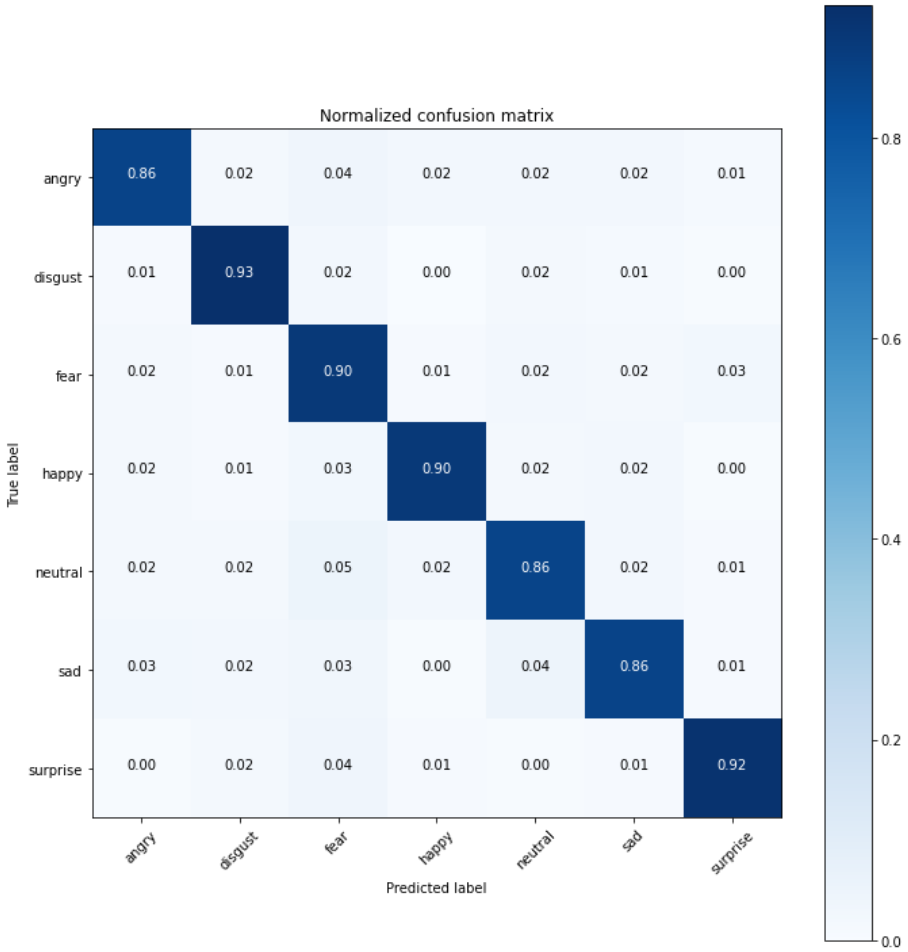


Figure50: confusion matrix for ResNet50 model during 50 epoch

Classification report

The classification report function shows numerical F-measures such as precision, recall, f1-score, and support for each class.

label	precision	recall	F-score
Angry	0.895	0.860	0.8775
Disgust	0.91	0.932	0.9209
Fear	0.808	0.896	0.8501
Happy	0.9294	0.896	0.9124
Neutral	0.8629	0.856	0.8595
Sad	0.8884	0.860	0.8739
Surprise	0.9349	0.920	0.9274

Table 12: Classification report from using ResNet50 model.

The classification report for ResNet50 shows the metric results comparing the actual classes to the predicted classes. The "surprise" and "Disgust" classes received the highest scores of approximately 92% for precision, recall, and F1 score. The "fear" emotion obtained the lowest score of 80% for all metrics.

3. DenseNet201 model

Metric results

Below is the plotted training history for model accuracy and model loss using the DenseNet201 model. The model was trained for 50 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

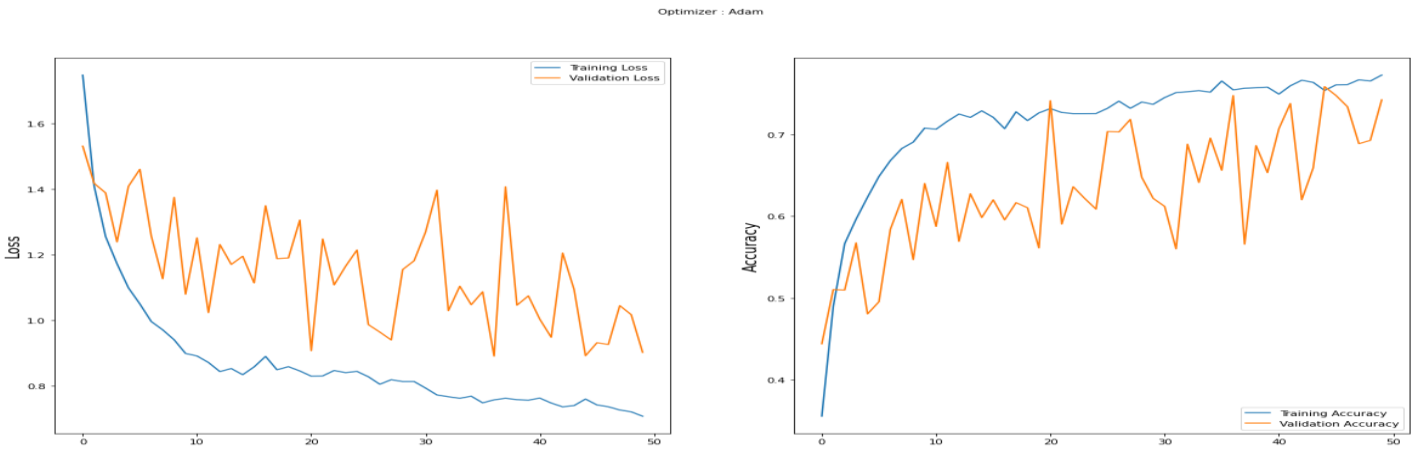


Figure 51: accuracy and loss history for DenseNet201 model

Confusion matrix

The confusion matrix obtained from training the DenseNet201 model for 50 epochs displayed a highest accuracy of 86% in predicting the 'surprise' class. However, the 'fear' class had the lowest accuracy of 58%.

The overall accuracy achieved by the DenseNet201 model was **74.46%.** While this indicates a moderate level of performance, it is lower compared to the other pretrained models previously evaluated.

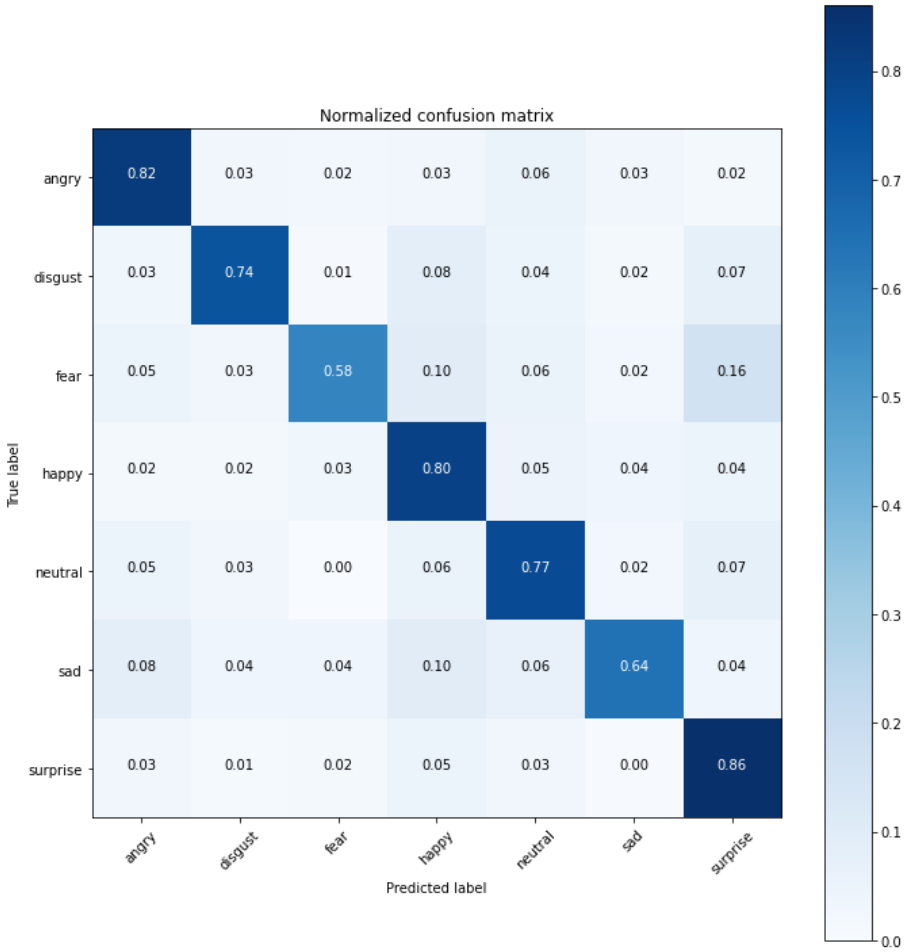


Figure 52: confusion matrix for DenseNet201 model during 50 epoch

Classification report

The classification report function shows numerical F-measures such as precision, recall, f1-score, and support for each class.

label	precision	recall	F-score
Angry	0.756	0.82	0.787
Disgust	0.83	0.74	0.782
Fear	0.833	0.58	0.684
Happy	0.66	0.8	0.723
Neutral	0.722	0.768	0.744
Sad	0.821	0.644	0.721
Surprise	0.678	0.86	0.758

Table 13: Classification report from using DenseNet201 model.

The classification report for DenseNet201 shows the metric results comparing the actual classes to the predicted classes. The "fear" and "sad" classes received the highest scores of approximately 82% for precision, recall, and F1 score. The "Happy" emotion obtained the lowest score of 66% for all metrics.

4. MobileNetV2 model (best model performance in real state)

Metric results

Below is the plotted training history for model accuracy and model loss using the MobileNetV2 model. The model was trained for 50 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

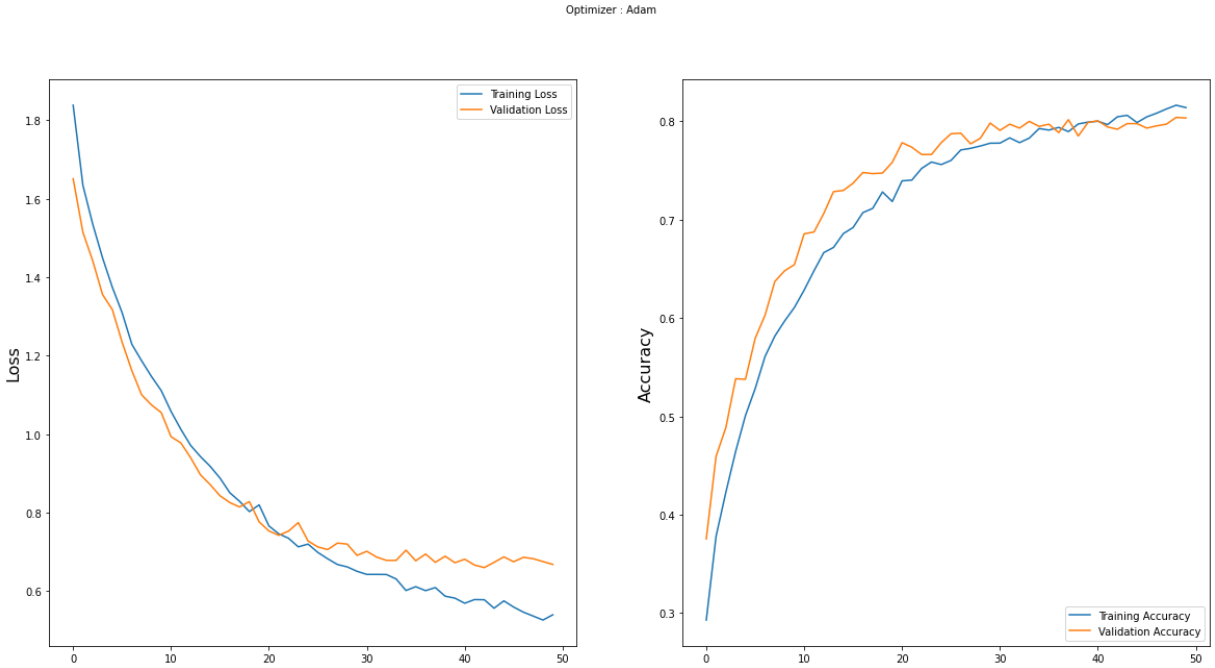


Figure 53: accuracy and loss history for MobileNetV2 model

Confusion matrix

The confusion matrix resulting from training the MobileNetV2 model for 50 epochs exhibited the highest accuracy of 87% in predicting the 'disgust' class. Moreover, the accuracy for the remaining classes hovered around 80%.

The overall accuracy attained was **80.86**%, indicating a significant level of success in accurately classifying emotions.

It is worth highlighting that the weights of the MobileNetV2 model were utilized due to their ability to deliver the best performance in real-world emotion prediction tasks. By leveraging these optimized weights, the model demonstrated superior performance in accurately identifying and classifying emotions in real-world scenarios.

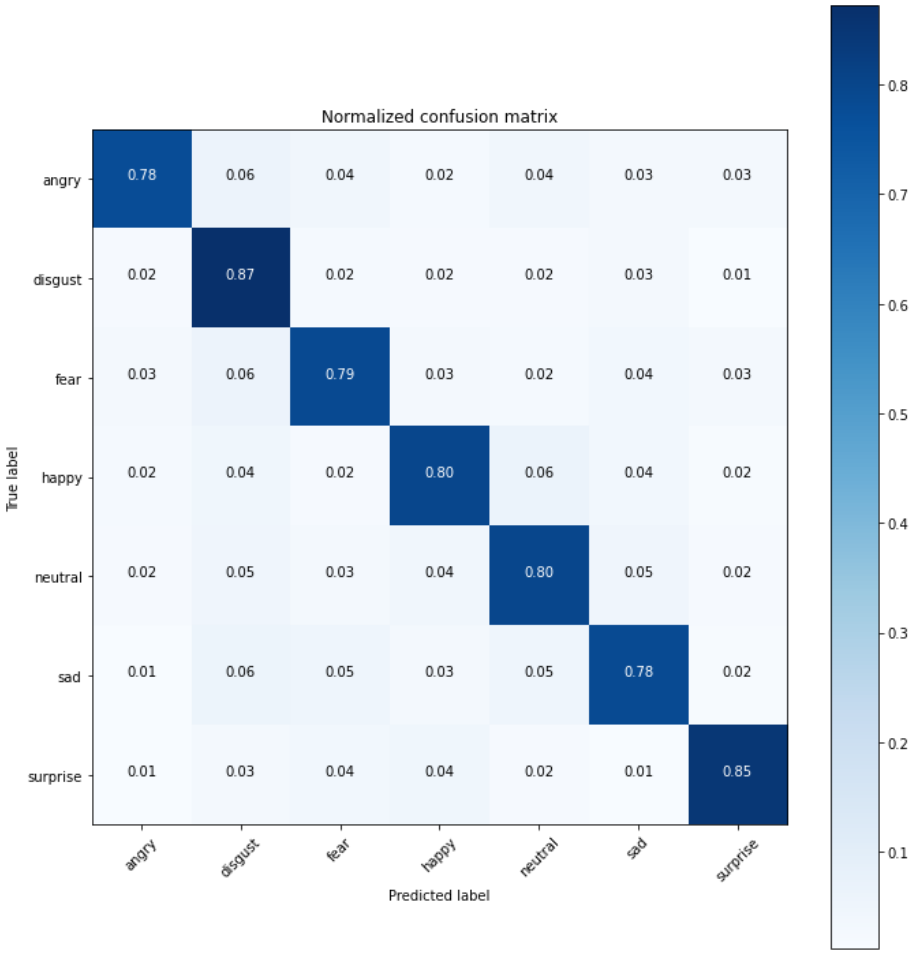


Figure 54: confusion matrix for MobileNetV2 model during 50 epochs

Classification report

The classification report function shows numerical F-measures such as precision, recall, f1-score, and support for each class.

label	precision	recall	F-score
Angry	0.866	0.776	0.818
Disgust	0.746	0.872	0.804
Fear	0.797	0.788	0.793
Happy	0.809	0.796	0.802
Neutral	0.783	0.796	0.789
Sad	0.803	0.784	0.793
Surprise	0.872	0.848	0.860

Table 14: Classification report from using DenseNet201 model.

The classification report for MobileNetV2 shows the metric results comparing the actual classes to the predicted classes. The "Angry" and "Surprise" classes received the highest scores of approximately 86% for precision, recall, and F1 score. The "Disgust" emotion obtained the lowest score of 74% for all metrics.

5. EfficientNetB0 model

Metric results

Below is the plotted training history for model accuracy and model loss using the EfficientNetB0 model. The model was trained for 50 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

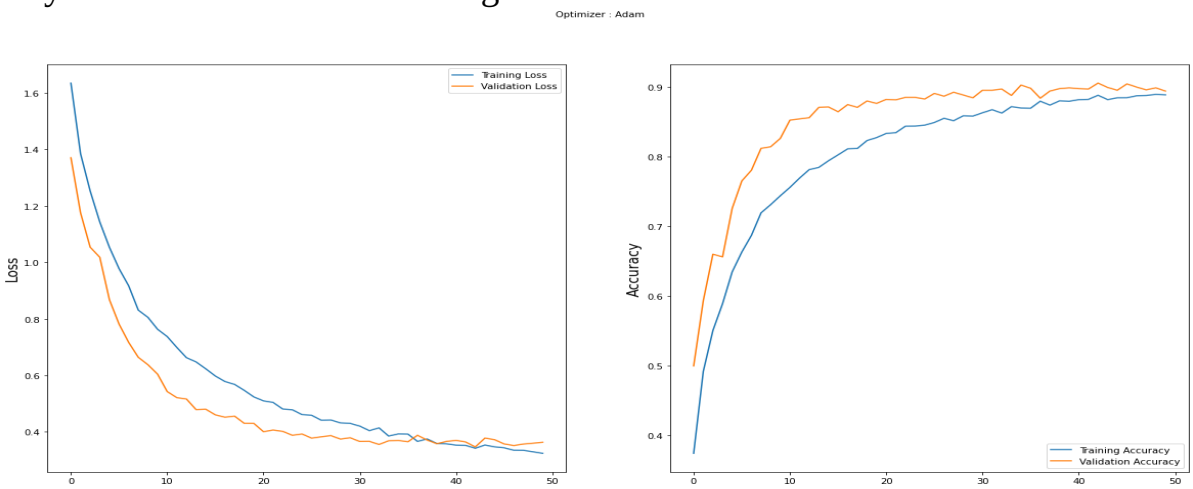


Figure 55: accuracy and loss history for EfficientNetB0 model

Confusion matrix

The confusion matrix obtained from training the EfficientNetB0 model for 50 epochs revealed a highest accuracy of 92% in predicting the 'surprise' and 'disgust' classes. Furthermore, the accuracy for the other classes remained consistently around 86%.

After training the EfficientNetB0 model for 50 epochs, it achieved an overall accuracy of **88.91**%. This indicates a high level of accuracy in accurately classifying emotions.

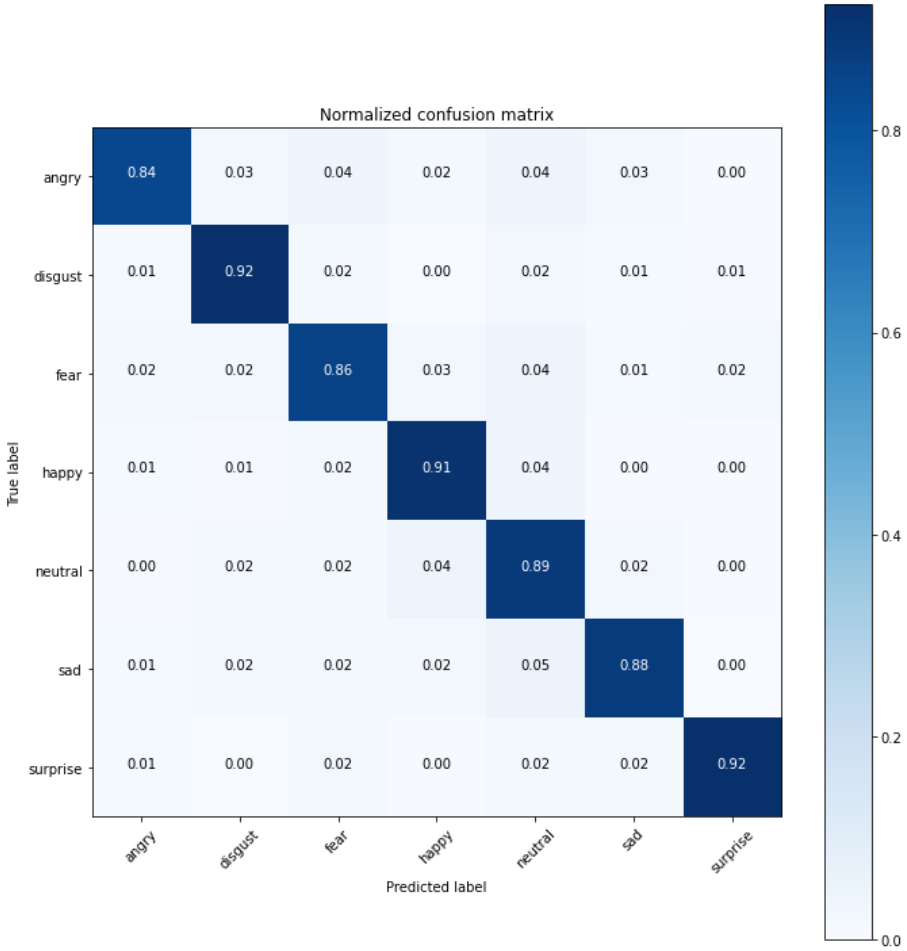


Figure 56: confusion matrix for EfficientNetB0 model during 50 epochs

Classification report

The classification report function shows numerical F-measures such as precision, recall, f1-score, and support for each class

label	precision	recall	F-score
Angry	0.925	0.84	0.8805
Disgust	0.909	0.924	0.9166
Fear	0.856	0.856	0.856
Happy	0.894	0.912	0.903
Neutral	0.798	0.888	0.8409
Sad	0.898	0.884	0.891
Surprise	0.958	0.92	0.9387

Table 15: Classification report from using EfficientNetB0 model.

The classification report for EfficientNetB0 shows the metric results comparing the actual classes to the predicted classes. The "Angry" and "Surprise" classes received the highest scores of around 92% for precision, recall, and F1 score. The "Neutral" emotion obtained the lowest score of 79% for all metrics.

6. EfficientNetB7 model (*Highest accuracy*)

Metric results

Below is the plotted training history for model accuracy and model loss using the EfficientNetB7 model. The model was trained for 50 epochs on a train dataset, and the accuracy and loss are shown in the figure below.

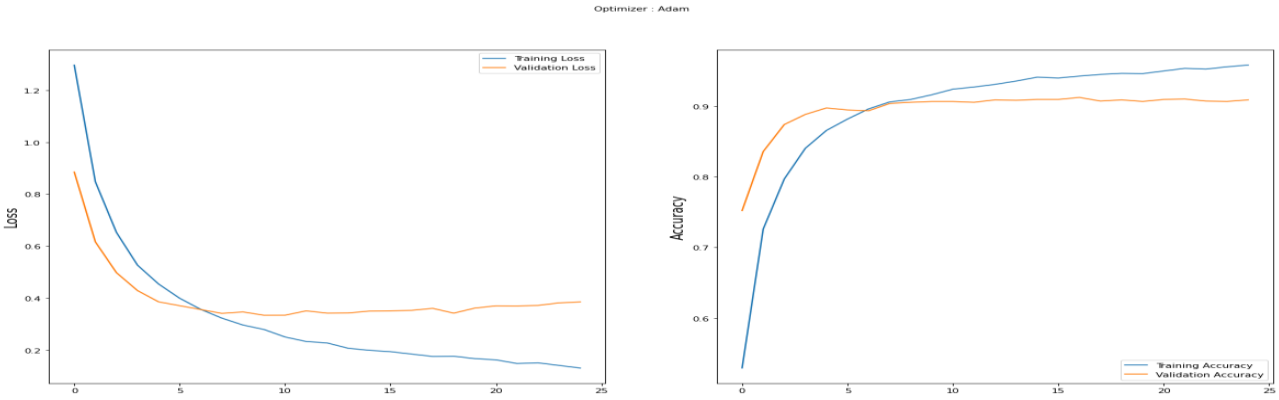


Figure 57: accuracy and loss history for EfficientNetB7 model

Confusion matrix

The confusion matrix resulting from training the EfficientNetB7 model for 50 epochs displayed a highest accuracy of 95% in predicting the 'surprise' class. Moreover, the accuracy for the remaining classes consistently hovered around 92%.

Among all the models evaluated, the EfficientNetB7 model demonstrated the highest accuracy of **90.91%** when trained for 50 epochs. However, it's important to note that while the model achieved a high accuracy score, its performance not translated well in real-world scenarios.

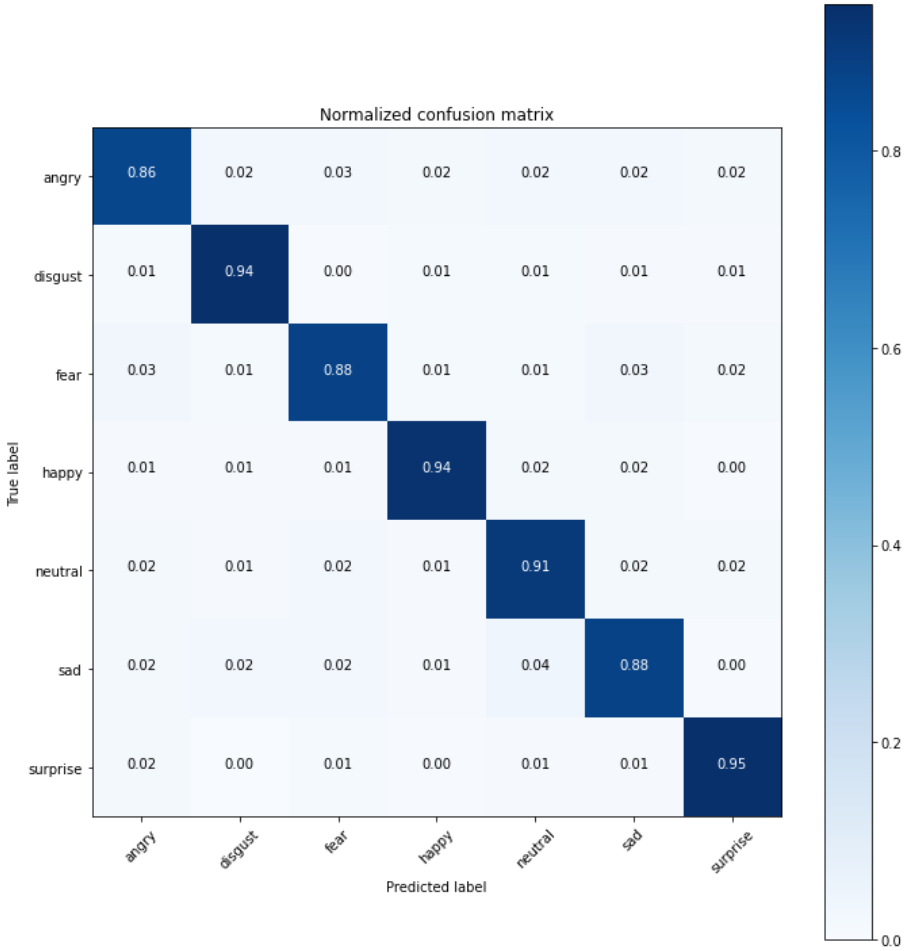


Figure 58: confusion matrix for EfficientNetB7 model during 50 epoch

Classification report

The classification report function shows numerical F-measures such as precision, recall, f1-score, and support for each class

label	precision	recall	F-score
Angry	0.885	0.864	0.874
Disgust	0.925	0.944	0.934
Fear	0.898	0.884	0.891
Happy	0.939	0.936	0.9378
Neutral	0.886	0.908	0.897
Sad	0.894	0.880	0.887
Surprise	0.933	0.948	0.940

Table 16: Classification report from using EfficientNetB7 model.

The classification report for EfficientNetB7 shows the metric results comparing the actual classes to the predicted classes. The "Happy" and "Surprise" classes received the highest scores of around 93% for precision, recall, and F1 score. The "Neutral" and "Angry" emotions obtained the lowest score of 88% for all metrics.

5.4. Predictions on unseen datasets

After the model has trained on the train dataset and receiving data of images and its corresponding labels, it was run on pictures from an unseen dataset. After applying the weights of the best evaluated model, EfficientNetB7, for prediction on 16 images, the model achieved remarkable results. It accurately predicted 15 out of the 16 images, with only one false prediction.
This high accuracy rate demonstrates the effectiveness of the EfficientNetB7 model in accurately classifying emotions in real-world images. The model's ability to correctly predict the majority of the images indicates its proficiency in capturing the essential features and patterns associated with different emotions.

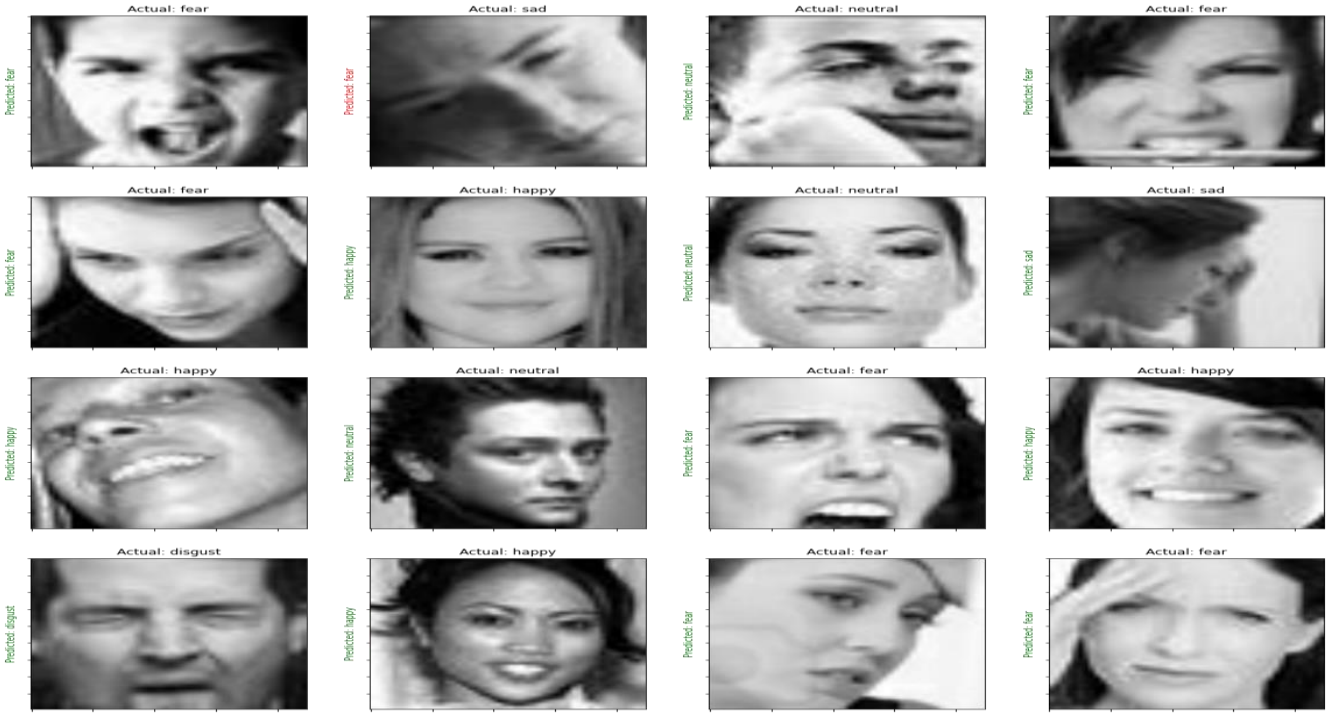


Figure 59: Predictions on unseen datasets

Predictions from testing

	Accuracy	Epochs
CNN before processing	64 %	80
CNN after processing	90.40%	50
VGG16	87.54%	50
ResNet50	88%	50
DenseNet201	74.46%	50
MobileNet	80.86%	50
EfficientNetB0	88.91%	50
EfficientNetB7	90.91%	50

Table 17: Predictions from testing

The table presents the accuracy of various models, with EfficientNetB7 achieving the highest accuracy of 90.91% after 50 epochs of training. However, its performance in real-world scenarios was comparatively weaker. In contrast, the MobileNetV2 model with pre-trained weights was employed and showcased superior performance in real-world emotion prediction tasks. By utilizing these optimized weights, the model achieved improved accuracy and adaptability.

Predictions Using Laptop Camera:
Before connecting the model with Esp32 camera and mobile application we use laptop camera for prediction as shown below.

Figure 60: model predictions using laptop

Charts:
Accuracy Chart:

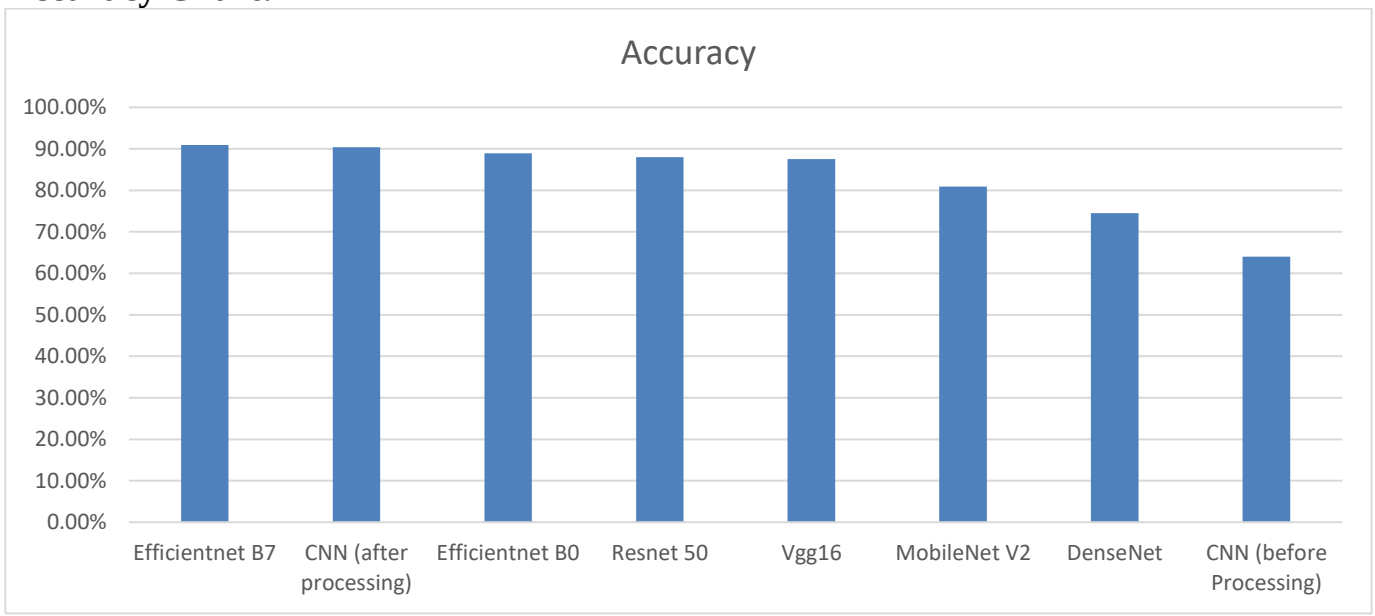


Figure 61: accuracy chart

Precision chart:

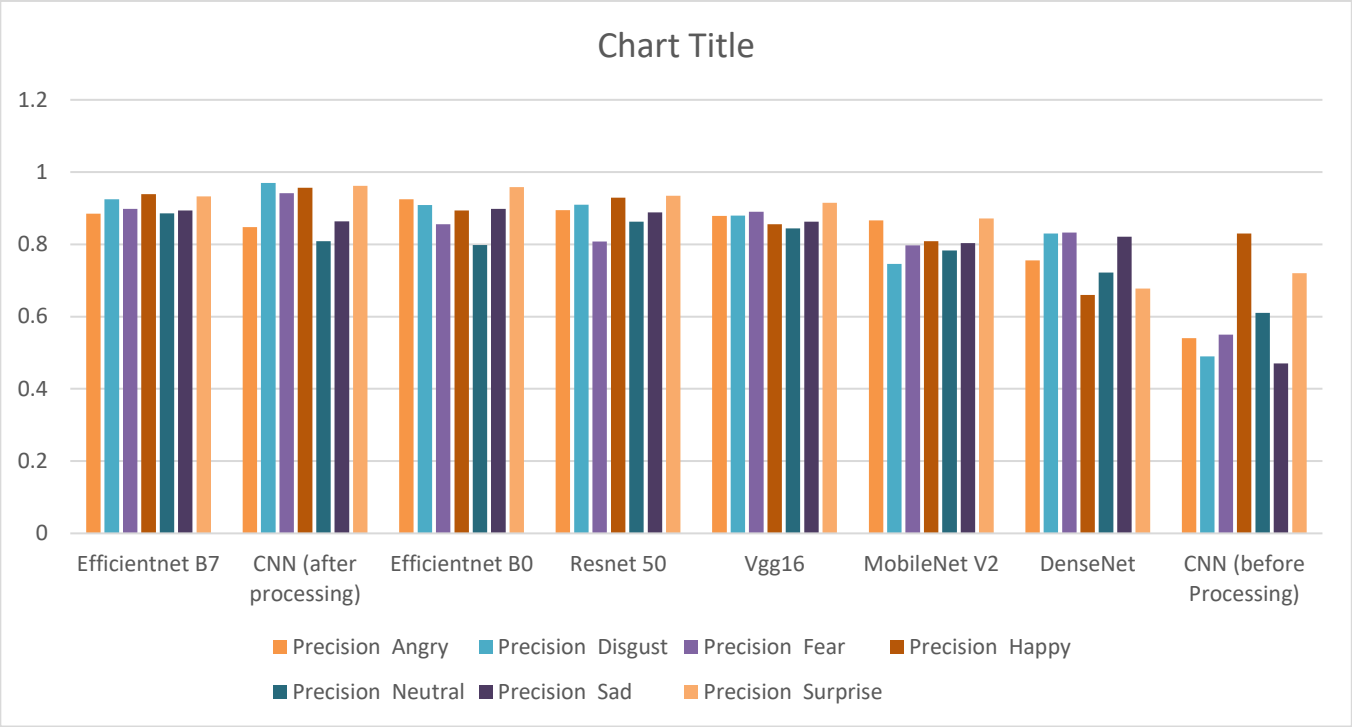


Figure 61: precision Chart

Recall Chart

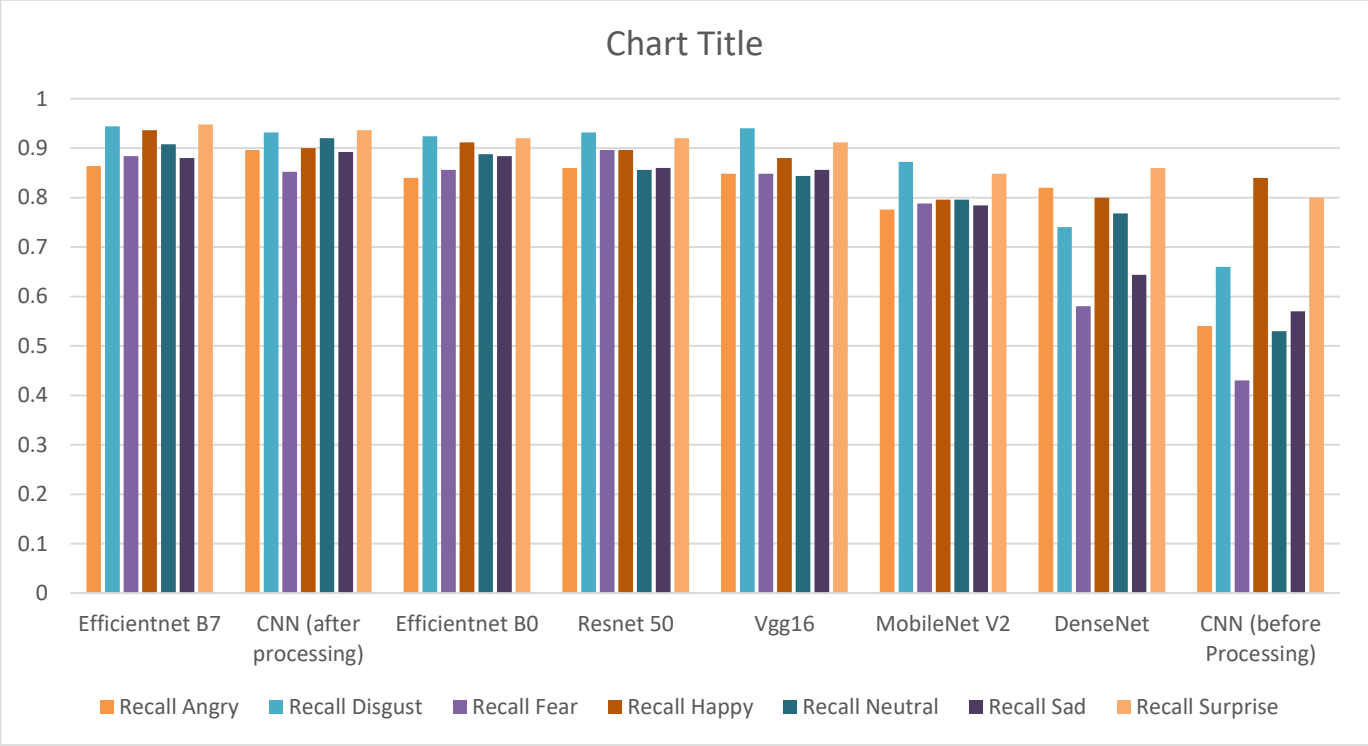


Figure 61: Recall chart

F-Score Chart

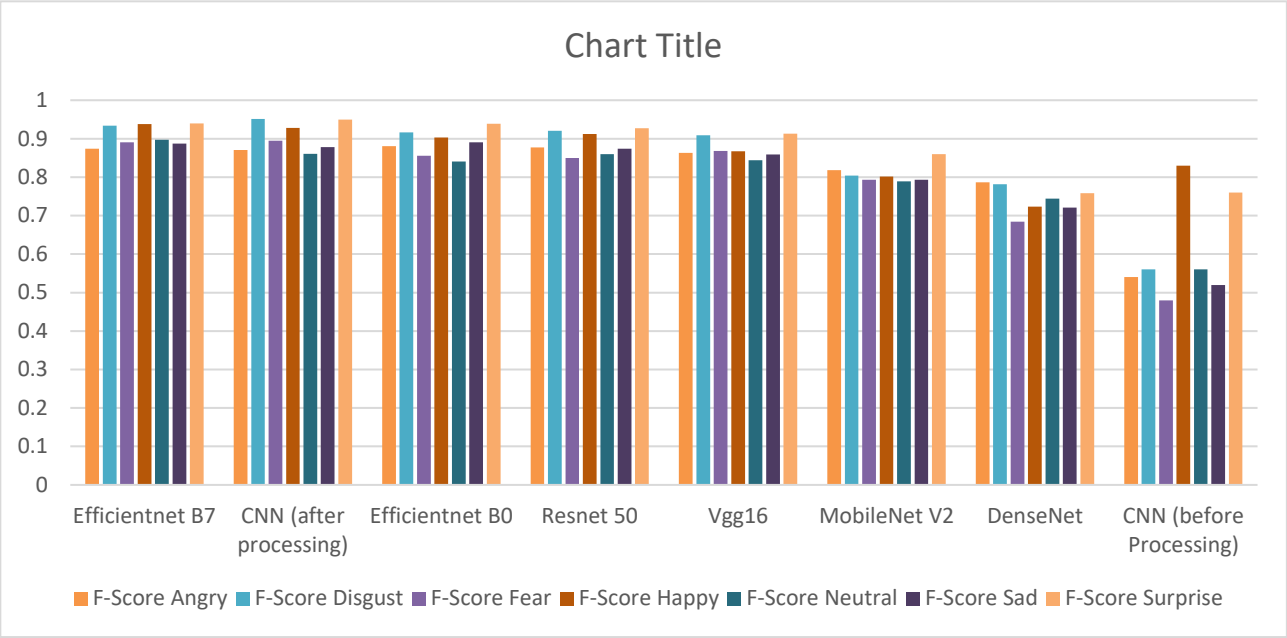


Figure 61: F-Score chart

5.5 mobile application

Figure 62: mobile application logo

5.5.1 Main activity

In our Android application, the main activity is configured to open on the home fragment. The home fragment serves as the initial screen that users see when they launch the app.

Upon launching the app, the main activity initializes and sets up the necessary components, including the navigation system and the fragment container. It then loads the home fragment into the fragment container, making it the visible content on the screen.

5.5.2 Home fragment

The home fragment is designed to display relevant information, interactive elements, and any other content that is essential for the user's first interaction with the app. It includes features such as image displays.

Firstly, it displays the output of the emotion detection model in real-time. When the model processes an image and detects the corresponding emotion, the home fragment dynamically presents the result to the user. This visual representation allows users to gain insights into the emotions captured by the model and enhances their understanding of the analyzed data.

In addition to the emotion detection output, the home fragment also incorporates the user's customization preferences. This means that when users apply settings or preferences within the application, such as choosing their gender or the type of images they prefer (cartoon, real, etc.), these choices are reflected in the home fragment. The fragment adapts its behavior and appearance based on the user's personalized settings, providing a tailored and unique experience.

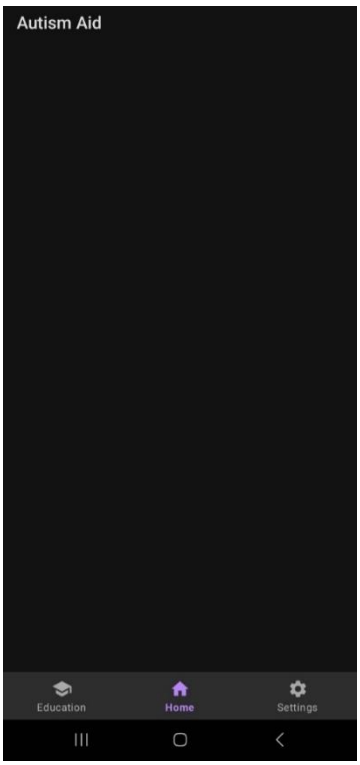


Figure 63: Home fragment

5.5.3 Setting fragment

In order to enhance the user experience within the app, a systematic approach was adopted by providing users with the ability to customize their preferences within the settings fragment, particularly in choosing.

1. Type

(Male or Female)

2. Image Type

(Cartoon, real, or Emojis).

3. Contact Button

4. Server Button

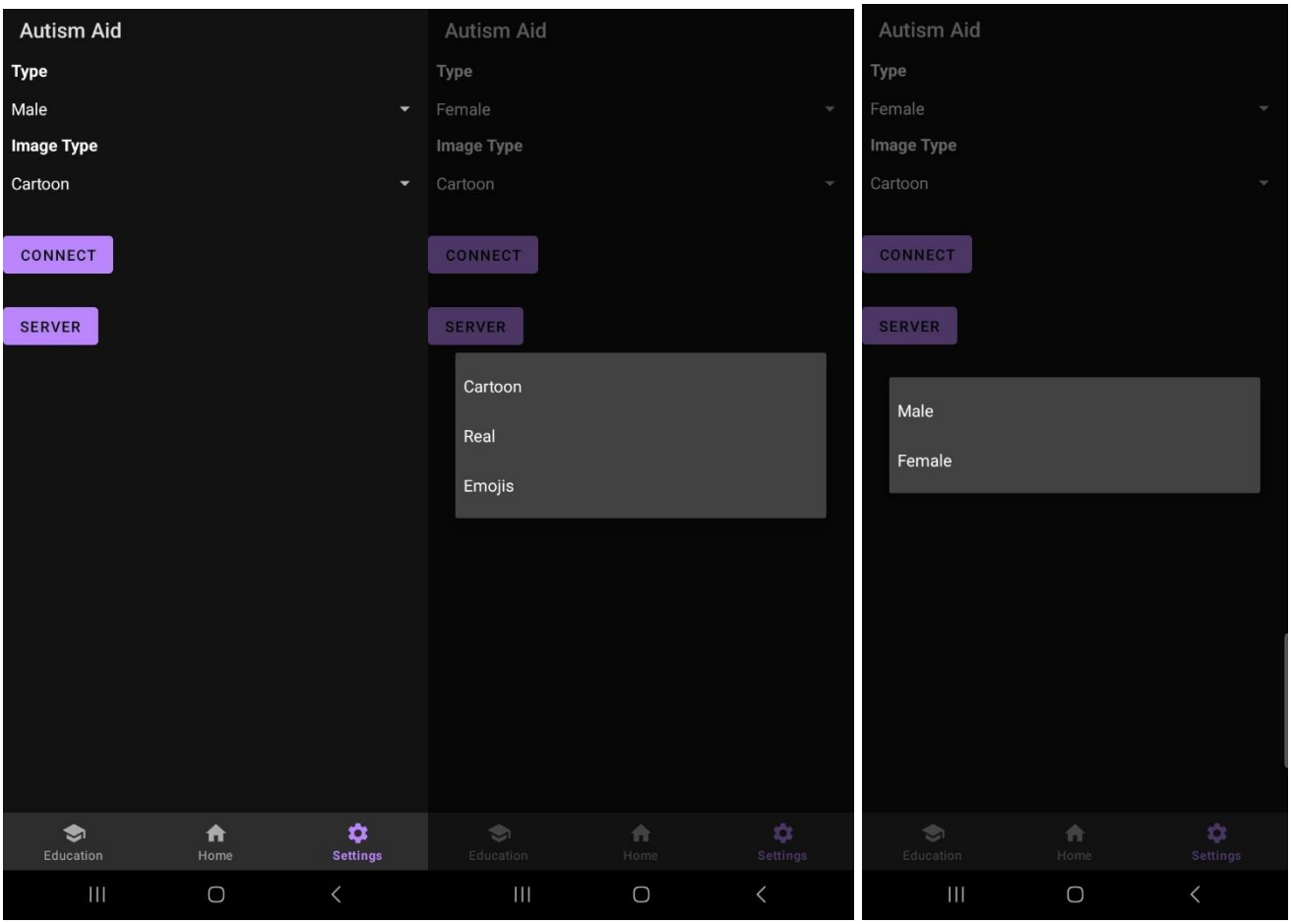


Figure 64: Setting fragment.

5.5.4 Education fragment

It is designed for educational purposes within the app. Although currently empty, it serves as a placeholder for future development and will be established in subsequent work.

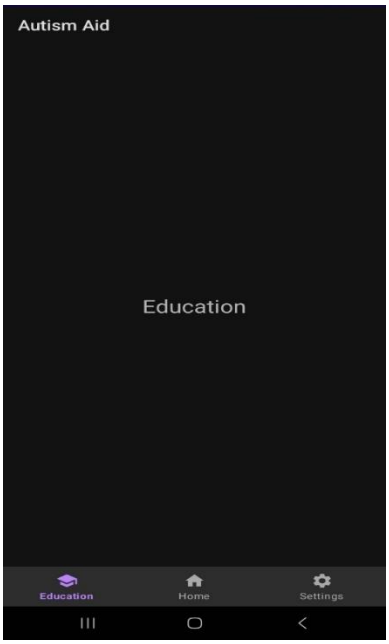


Figure 65: Education fragment

5.5.5 Problems and Solutions

Problem	Solutions
Data Unbalancing	Try SMOTE up sampling and down sampling
	GAN but it isn't work well need many images.
	Under sampling to disgust class then Up sampling by augmentation
Saving model	EfficientNet not work with model.save() function so, we use another pretrained model like Vgg 16 and the CNN but it wasn't performed well with real-time capturing and MobileNet was the best model for real-time capturing.
Open laptop camera after loading the model	The prediction does not change the prediction label but after using another way of saving ModelCheckpoint callback it saves the correct weights and then the labels changes with changing the facial expressions
Bluetooth library (pybluez)	Bluetooth library was hard to import but after searching that install Microsoft Visual C++ Build Tools then pybluez package is installed.
Using Bluetooth on mobile application didn't work	After searching we used Bluetooth on mobile application
Esp32 camera not compatible with OpenCv library	By taking pictures by esp32 camera then upload it to python code the OpenCv worked.

Table 18: the problem and its solution

Chapter 6

Conclusion & Future work

6.1 Conclusion

In conclusion, this project represents a significant contribution to the ongoing efforts to address the challenges faced by individuals with ASD. By leveraging deep learning algorithms and transfer learning techniques, we have achieved notable improvements in the performance and accuracy of the models used for emotion recognition. EfficientNetB7 achieved the highest accuracy of 90.91% after 50 epochs of training, but its performance in real-world scenarios was comparatively weaker. On the other hand, the MobileNetV2 model with pre-trained weights showcased superior performance in real-world emotion prediction tasks, demonstrating its reliability and adaptability.

We have also developed a mobile application consisting of three fragments: home, education, and setting. The settings fragment allows users to customize their preferences, providing a tailored and unique experience. The home fragment dynamically presents the results of the emotion detection model and adapts its behavior and appearance based on the user's personalized settings, enhancing the user experience.

The innovative hardware design, as represented by the wearable glasses, seamlessly integrates with the software, forming a comprehensive solution that empowers individuals with ASD in their social interactions and educational endeavors. The combination of hardware and software advancements has the potential to revolutionize the support and empowerment of individuals with ASD, substantially improving their quality of life and fostering their inclusive participation in society.

6.2 Future work

We will establish an education fragment within the app to cater to the educational needs of individuals with autism. The Education fragment aims to provide specific features and resources that support learning and development for individuals on the autism spectrum.

Additionally, improving the accuracy of the FER model and continually adding new features and functionalities based on user feedback and research findings can enhance the app's effectiveness in supporting individuals with autism. By combining advanced deep learning techniques, utilizing domain-specific datasets, and adopting a user-centered design approach, we can continuously enhance the performance, accuracy, and usability of the app to provide meaningful educational support for individuals with autism.

References

1. Jonathan Pevsner, ‘BIONFORMATICS AND FUNCTIONAL GENOMICS’, third edition

1. Autism Spectrum Disorder. (n.d.). National Institute of Mental Health (NIMH). https://www.nimh.nih.gov/health/topics/autism-spectrum-disorders-asd#:~:text=Autism%20spectrum%20disorder%20(ASD)%20is,first%20two%20years%20of%20life

2. Yamashita, R. *et al.* (2018) *Convolutional Neural Networks: An overview and application in radiology - insights into imaging, SpringerOpen*. Available at: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9.

3. Facial Emotion Recognition (FER) is a technology that analyzes facial expressions in images to reveal information about a person's emotional state. Researchers from a variety of sectors are becoming increasingly interested in FER as it has a wide range of applications, but of special importance is human-computer interaction.

4. Yamashita, R. *et al.* (2018) *Convolutional Neural Networks: An overview and application in radiology - insights into imaging, SpringerOpen*. https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9 .

5. Team, K. (no date) *Keras Documentation: Transfer Learning & Fine-tuning, Keras*.

6. Hanna, K. T., & Wigmore, I. (2023, February 13). *What is a mobile app (mobile application)? – TechTarget definition*. WhatIs.com. https://www.techtarget.com/whatis/definition/mobile-app

7. *What are the different types of mobile apps? and how do you choose?* (no date) *CleverTap*. Available at: https://clevertap.com/blog/types-of-mobile-apps/.

8. *The difference between frontend and backend mobile app development* (2022) *Yellow*.
9. *Android App Development Fundamentals for Beginners* (2022) *GeeksforGeeks*.

10. says:, T. *et al.* (no date) *XML in Android: Basics and different XML files used in Android, Abhi Android*.

11. Wu, C.-D. and Chen, L.-H. (2019) *Facial emotion recognition using Deep Learning, arXiv.org*.

12. Khaireddin, Y. and Chen, Z. (2021) *Facial emotion recognition: State of the art performance on FER2013*.

13. ResNet-50 and VGG-16 for recognizing Facial Emotions Poonam Dhankhar1 1Department of Computer Science and Engineering, MSIT, New Delhi, India 2021

14. *Facial emotion recognition using Deep Learning - oda.oslomet.no*.

15. Sena, D. (2023) *CK+ dataset, Kaggle*.

16. Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). AffectNet: A database for facial expression, valence, and arousal computing in the wild. IEEE Transactions on Affective Computing, 10(1), 18-31.

17. Sambare, M. (2020) *Fer-2013*, *Kaggle*.

18. Kamat, S. (2023). How To Use GAN To Generate Images. *Data, AI & Product Engineering | Algoscale*. https://algoscale.com/blog/how-to-use-gan-to-generate-images/#:~:text=GANs%20use%20two%20neural%20networks,voice%20generation%2C%20and%20video%20generation

19. Korstanje, J. (2022, January 5). SMOTE | Towards Data Science. *Medium*.

20. Ray, S. (2022, January 4). What Is Data Augmentation? - Lansaar - Medium. *Medium*. https://medium.com/lansaar/what-is-data-augmentation-3da1373e3fa1

21. Kharwal, A. (2021). Convert Image to Array using Python. *Thecleverprogrammer*.

22. *What is Normalization in Machine Learning | Deepchecks*. (2021, August 5). Deepchecks.

23. Aslan, Z. (2022, March 16). Transfer Learning Using DenseNet201 - MLearning.ai - Medium.

24. Boesch, G. (2023). Deep Residual Networks (ResNet, ResNet50) – 2023 Guide. *viso.ai*.

25. Team, K. (n.d.). *Keras documentation: VGG16 and VGG19*. https://keras.io/api/applications/vgg/

26. Sarkar, A. (2023, May 19). Building MobileNet from Scratch Using TensorFlow - Towards Data Science.

27. Gurion, D. B. (2022, January 7). Image Classification Transfer Learning and Fine Tuning using TensorFlow.

28. Kulkarni, A., Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy. In *Elsevier eBooks* (pp. 83–106).

29. Descending into ML: Training and Loss. (n.d.). *Google for Developers*.

30. D, E. (2021, December 12). Accuracy, Recall & Precision - Erika D - Medium. *Medium*. https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d

31. Shung, K. P. (2020, April 10). Accuracy, Precision, Recall or F1? - Towards Data Science. *Medium*. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

32. *Raspberry Pi Based Emotion Recognition using OpenCV, TensorFlow, and Keras*. (n.d.). https://circuitdigest.com/microcontroller-projects/raspberry-pi-based-emotion-recognition-using-opencv-tensorflow-and-keras

33. GeeksforGeeks. (2022). Android App Development Fundamentals for Beginners. *GeeksforGeeks*. https://www.geeksforgeeks.org/android-app-development-fundamentals-for-beginners/

34. Gattal, A. (2018b, August 26). Understand Android Basics Part 1: Application, Activity and Lifecycle. *Medium*. https://medium.com/@Abderraouf/understand-android-basics-part-1-application-activity-and-lifecycle-b559bb1e40e

35. Özkoç, H. (2022, January 5). Android activity and lifecycle example | Medium. *Medium*.

-GeeksforGeeks. (2021). Activity Lifecycle in Android with Demo App. *GeeksforGeeks*. https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/

36. GeeksforGeeks. (2022a). Android UI Layouts. *GeeksforGeeks*.

37. *Linear Layout Tutorial With Examples In Android | Abhi Android*. (n.d.).

38. Spinners. (n.d.). *Android Developers*.

39. Pathak, A. (2023, March 14). Fragments in Android - Abhishek Pathak - Medium. *Medium*.

40. Hakeem, H. (2021, December 13). Android Fragments: FragmentContainerView - ProAndroidDev.

41. Smith, R. (2018, July 17). Switching Between Fragments Without the Mindless Killing Spree.

42. Suleiman. (2019, November 30). *Bottom Navigation Bar with Fragments - Android Tutorial*. Suleiman's Blog. https://blog.iamsuleiman.com/bottom-navigation-bar-android-tutorial/

43. *BluetoothSocket ： android developers Android Developers*.

44. Design: html5up.net / Modified: Mike Petrichenko. (n.d.). *Bluetooth Framework and RFCOMM Protocol*. Copyright (C) 2006-2023 Mike Petrichenko, Soft Service Company. \

45. Run apps on the Android Emulator. (n.d.). *Android Developers*.

46. Penumudy, T. (2021, December 25). Computer Vision and Image Processing with OpenCV - Analytics Vidhya - Medium.

47. Mahatatineni. (2021, December 16). Haar Cascade Classifier - Mahatatineni - Medium. *Medium*. https://medium.com/@mahatatineni.931/haar-cascade-classifier-3b6b4df69d39

48. Mittal, A. (2021, December 24). Haar Cascades, Explained - Analytics Vidhya - Medium.

49. GeeksforGeeks. (2021). Activity Lifecycle in Android with Demo App. *GeeksforGeeks*.

50. *ESP32-CAM Development Board - With Camera - RAM Electronics*. (2021, November 13). RAM Electronics. https://ram-e-shop.com/product/esp32-cam-development-board-with-camera/

52. *UNO R3 | Arduino Documentation*. (n.d.). https://docs.arduino.cc/hardware/uno-rev3.