A Wearable Behavioral Aid for Autistic Children

Our team

Abdulrhman Nasser

Merna Ahmed Mansour

Shaimaa Abd Elkhalek

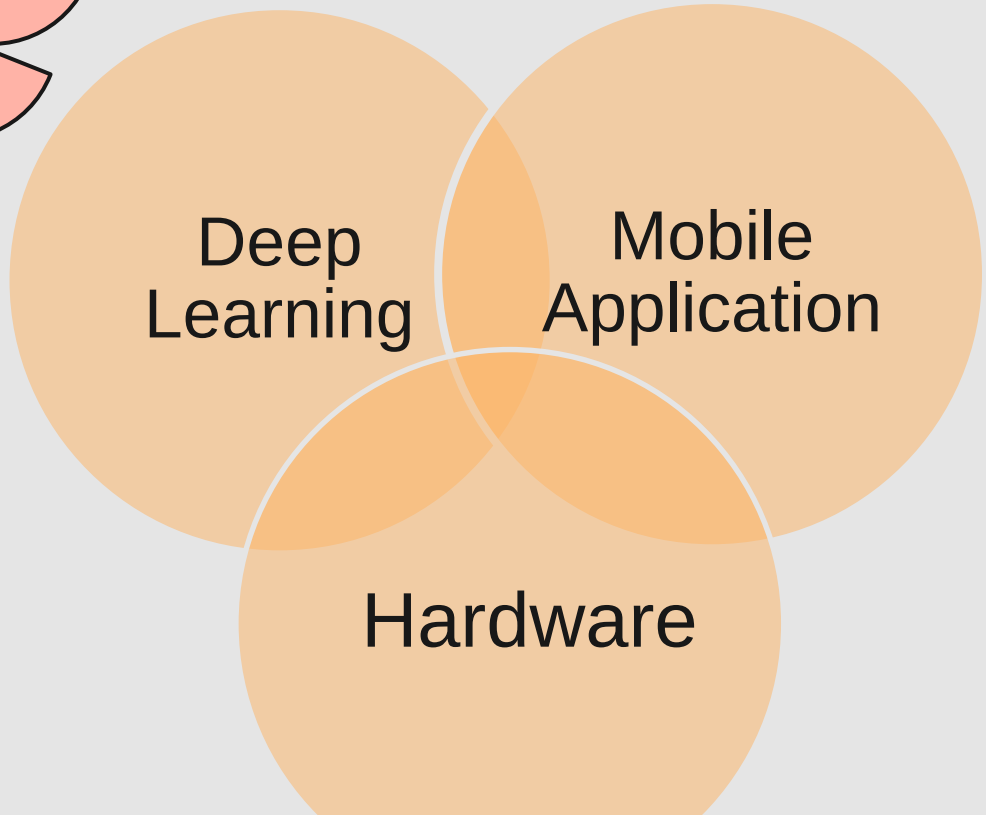
Supervised by:
Prof. Ibrahim Sadik

Table of contents

01
Introduction

What is our project?

Wearable behavioral aid for autistic children



What is Autism?

Autism is a lifelong developmental disability which affects how people communicate and interact with the world. Caused by differences in the brain.

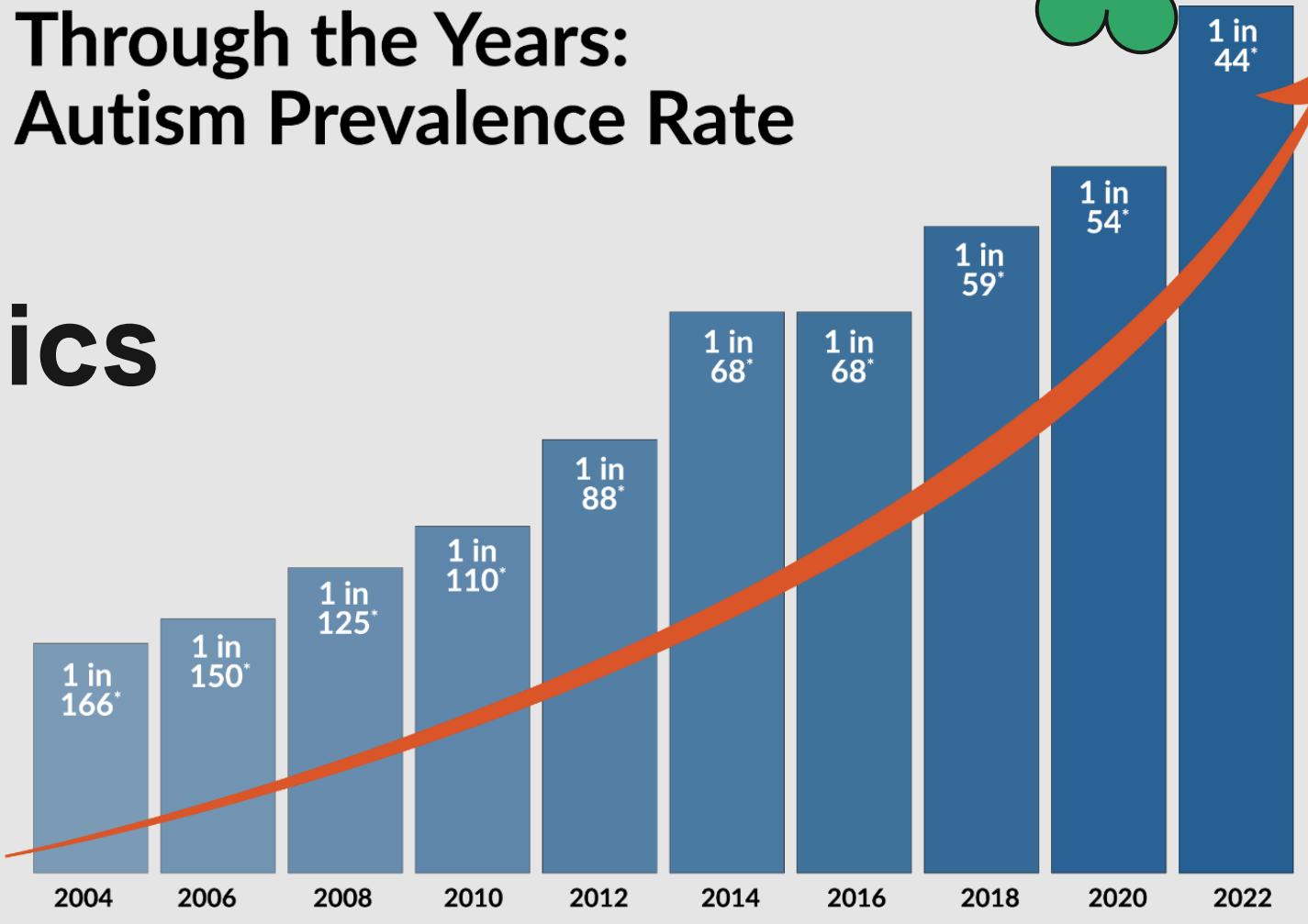
Signs

There are many signs of autism, and they include:

- Avoiding or does not keep eye contact.
- Finding it hard to understand what others are thinking or feeling getting very anxious about social situations.
- Finding it hard to make friends or preferring to be on your own.

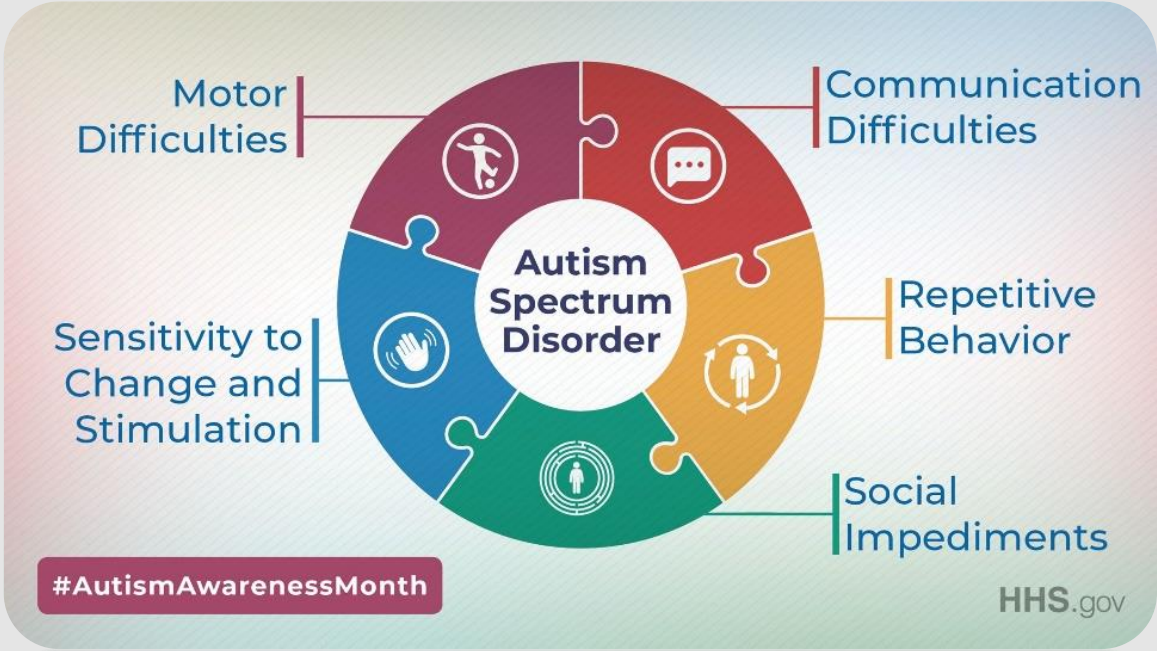
Statistics

Through the Years: Autism Prevalence Rate



Target Segmentation

Any patient that falls under the category of social impediments & communication difficulties and can use a mobile



02
MOTIVATION & OBJECTIVES

Motivation

- Most adults on autism spectrum want to have friends and a social life, however, they just don't know how.
- And this is due to the fact that people don't understand adults on autism spectrum.
- As it's hard for adults with autism to communicate their feelings and understand the feelings of others.

Motivation

They are usually misunderstood for being rude and casted out of social activities.

Which in turn increases their social anxiety making it hard for them to blend in their societies.

Objective

- Our goal is to decrease the gap between the autistic people and others. And this is done by facilitating a mean of communication between the two. This communication bridge is built via our A wearable behavioral aid for autistic children

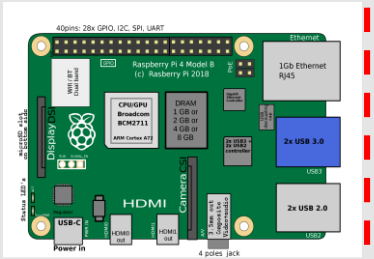
Objective

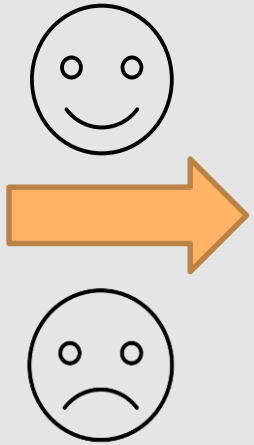
“I am not able to talk out of my mouth, however I have found another way to communicate by spelling on my computer.”- Carly Fleishmann

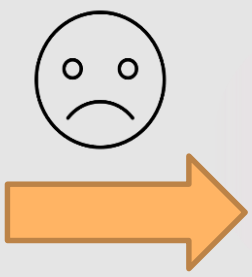
03
Block Diagram

Block Diagram

why are you sad I'm here for you







Project funding

- Information Technology Industry Development Agency (ITIDA).
- The project is funded by 8000 L.E.

04

Framework(tools)

- Python
- Deeplearning
- Datasets from Kaggle
- TensorFlow
- Keras
- OpenCV
- Java
- Android studio
- Gennymotion
- emulator android studio

kaggle

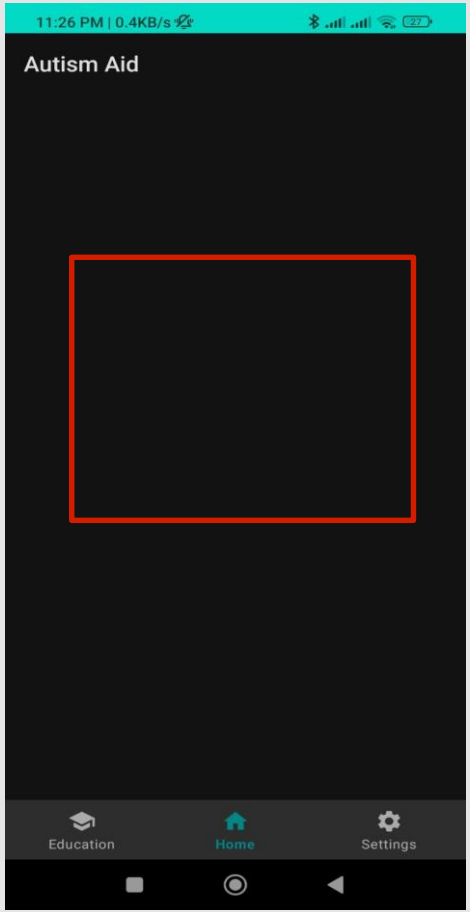
05
Mobile Application

Mobile Application

MainActivity contains:

- Navigation bar
- Fragment Container

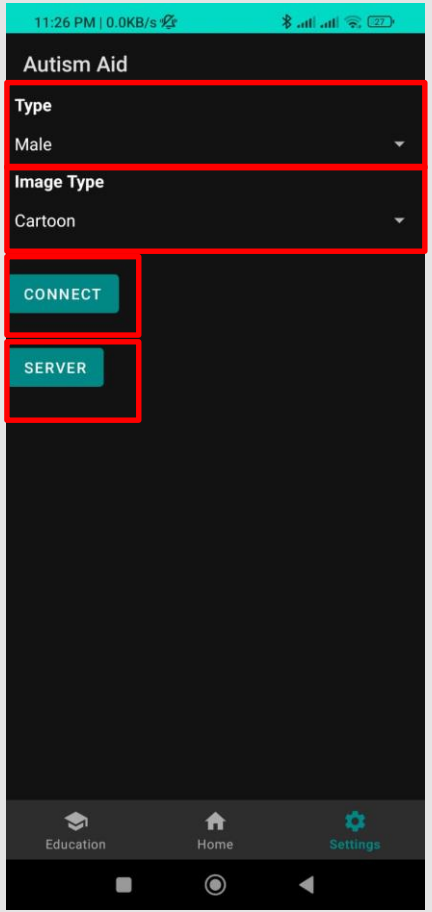
Mobile Application



Home Fragment contains:

- Image View

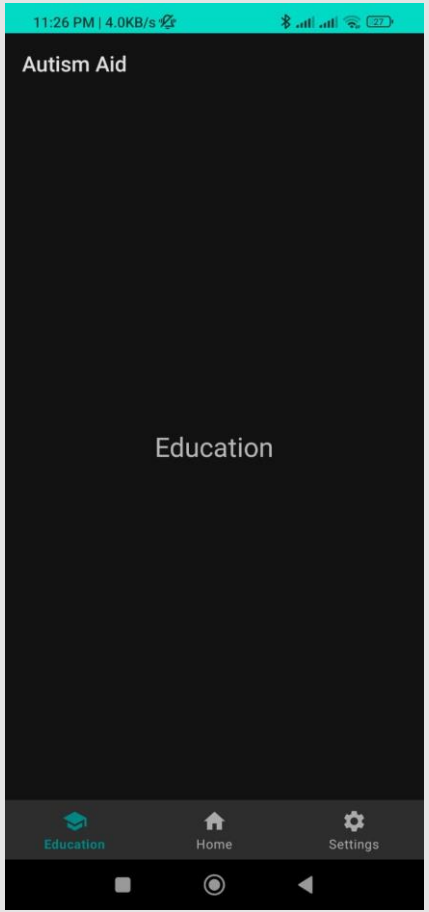
Mobile Application



Settings Fragment contains:

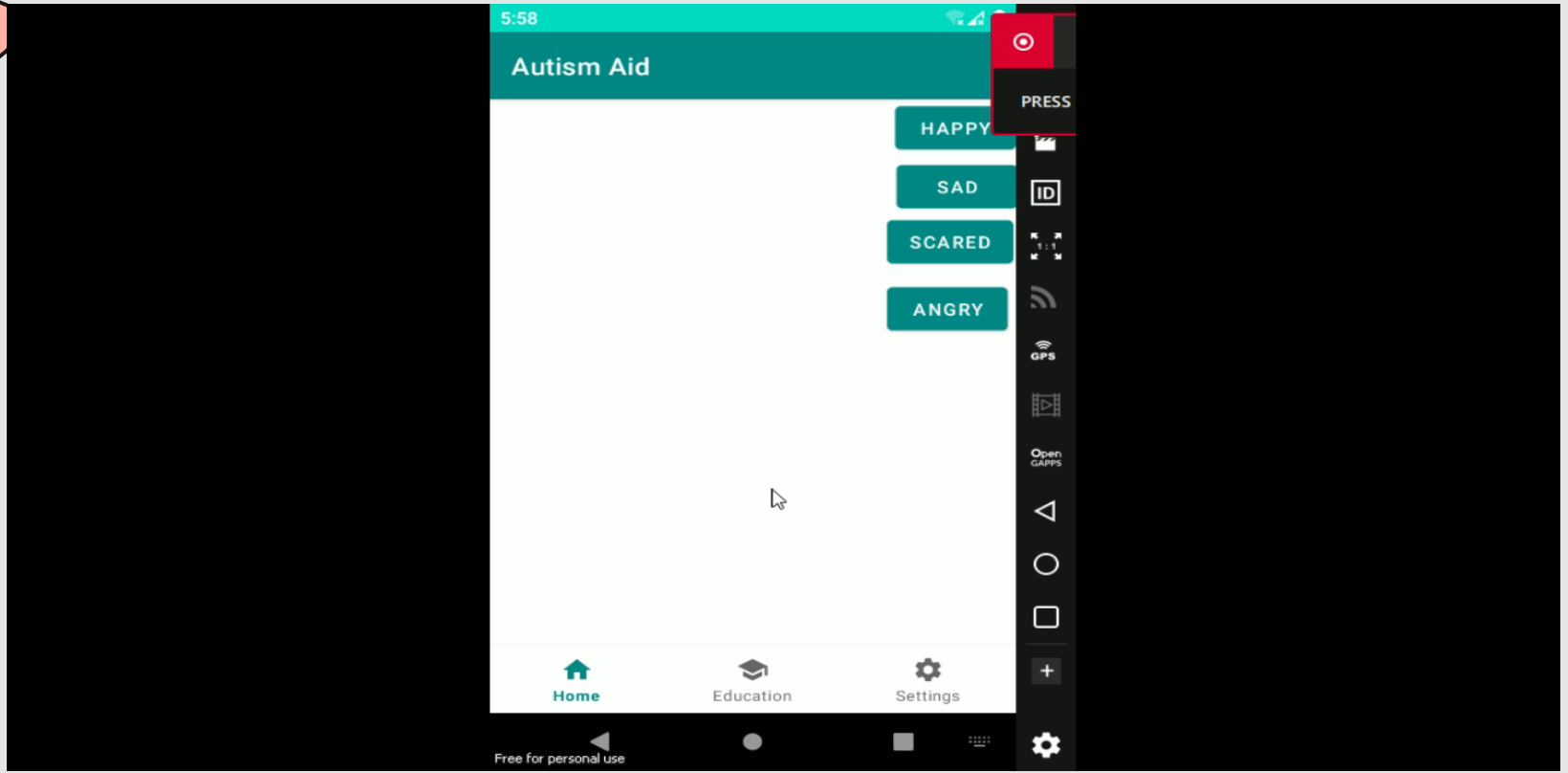
- Type Spinner
- Image type spinner
- Connect button
- Server button

Mobile Application

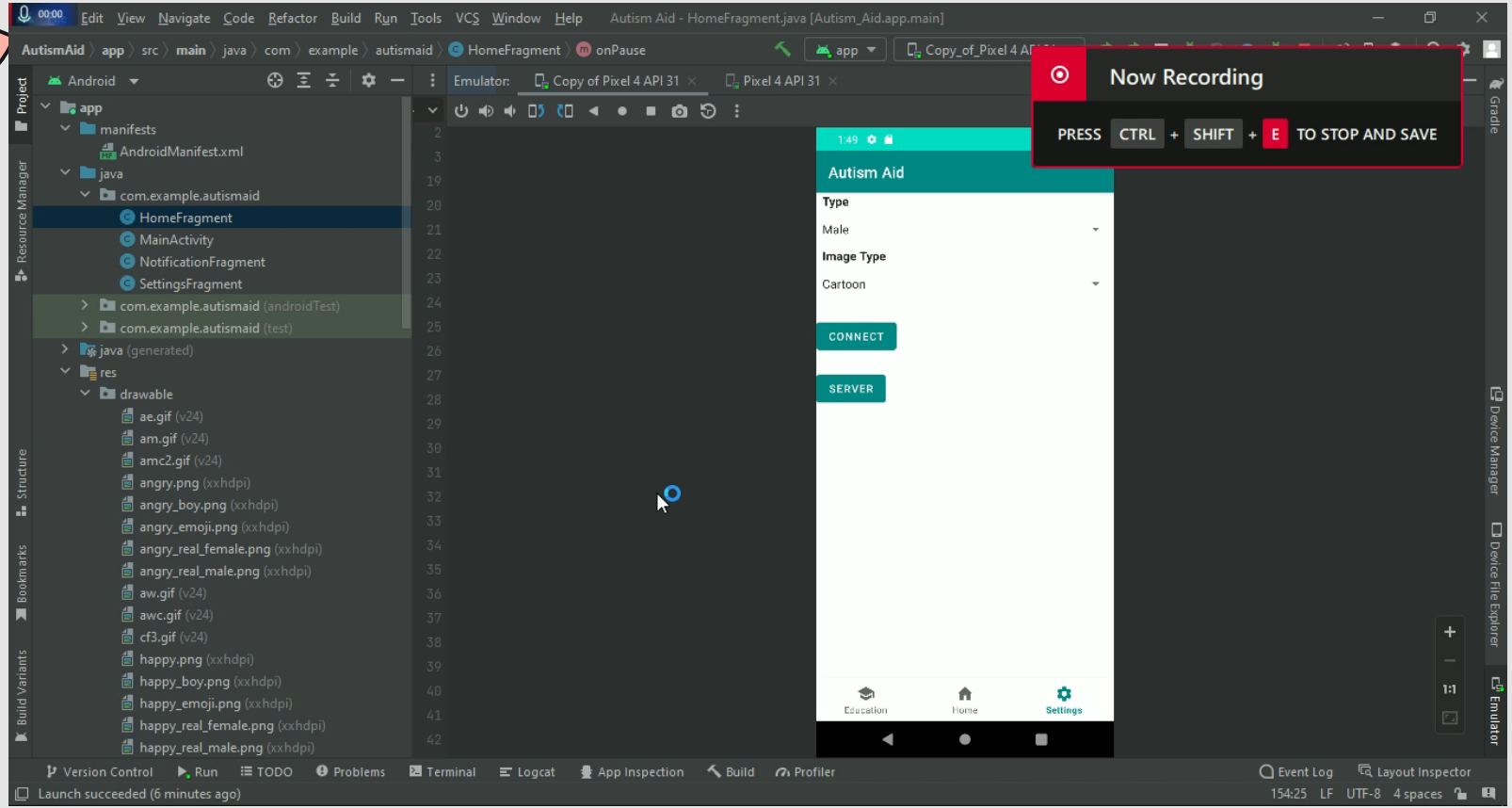


Education Fragment is a place holder for future installments according to the doctor that buys the product.

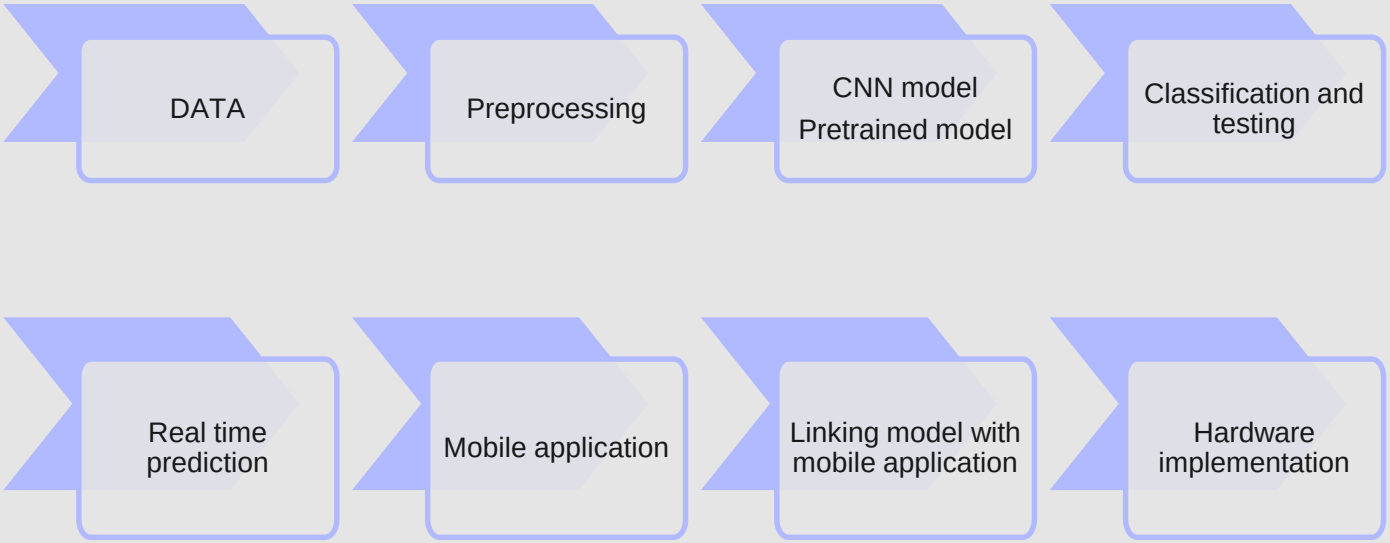
Mobile Application



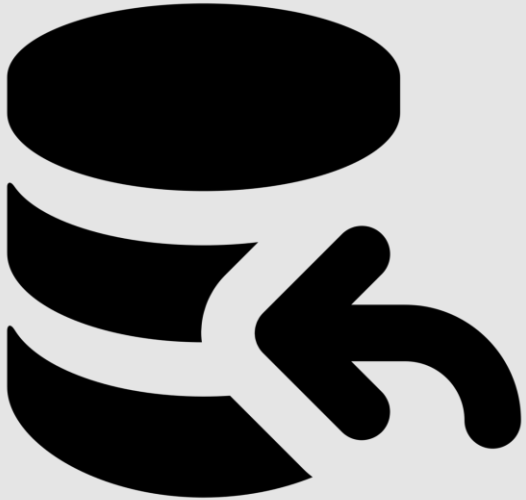
Mobile Application



06
Workflow

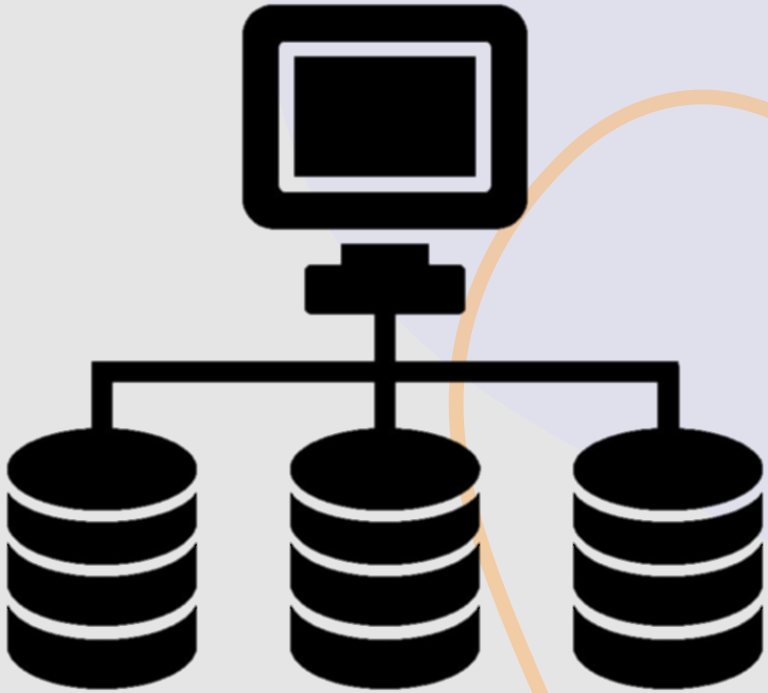

DATA
Preprocessing
CNN model
Pretrained model
Classification and testing
Real time prediction
Mobile application
Linking model with mobile application
Hardware implementation

07
DATASET



Datasets

- Ck+
- AFFECTNET
- FER2013



Datasets

	Ck+	FER2013	AFFECTNET
Anger	45	4953	28,130
Disgust	59	547	5,264
Fear	25	5121	8,191
Happiness	69	8989	146,198
Sadness	28	6077	29,487
Surprise	83	4002	16,288
Neutral	593	6198	80,276
Contempt	18	0	5,135
Overall	920	35,887	1,000,000

Datasets (CK+)

- grayscale Smallest data set and augmented but has the highest accuracy and precision
- Contains data **up to 920 images from 920 original CK+ dataset**
- Data is already reshaped to **48x48 pixels**, in **grayscale format Emotions are defined as determined index below:**

Affectnet

- Affectnet is a large facial expression dataset that contains over one million images
- manually labeled for the presence of eight facial expressions along with the intensity of valence and arousal.

- But it takes long time processing and after balancing classes gives small accuracy and can't predict well
- It gives low accuracy (55.60%)

Number of Annotated Images in Each Category

Expression	Number
Neutral	80,276
Happy	146,198
Sad	29,487
Surprise	16,288
Fear	8,191
Disgust	5,264
Anger	28,130
Contempt	5,135
None	35,322
Uncertain	13,163
Non-Face	88,895

FER2013

- The training set consists of 28,709 examples and the public test set consists of 3,589 examples
- The data consists of 48x48 pixel grayscale images of faces. The faces have been
- Imbalanced dataset
- The smallest class is disgust (547 images)
- The largest class is happy(8989)

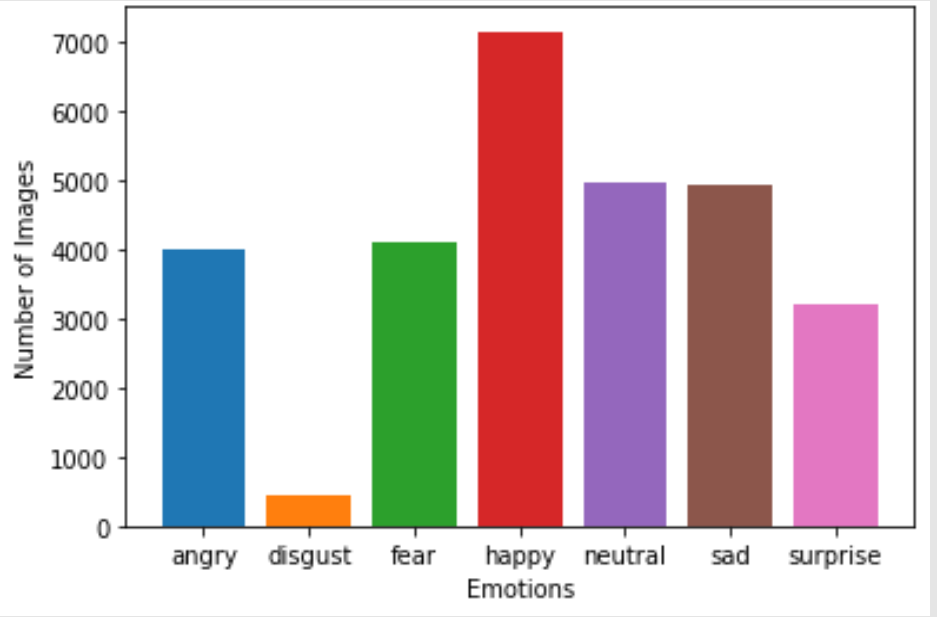
Preprocessing

Datasets problems

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

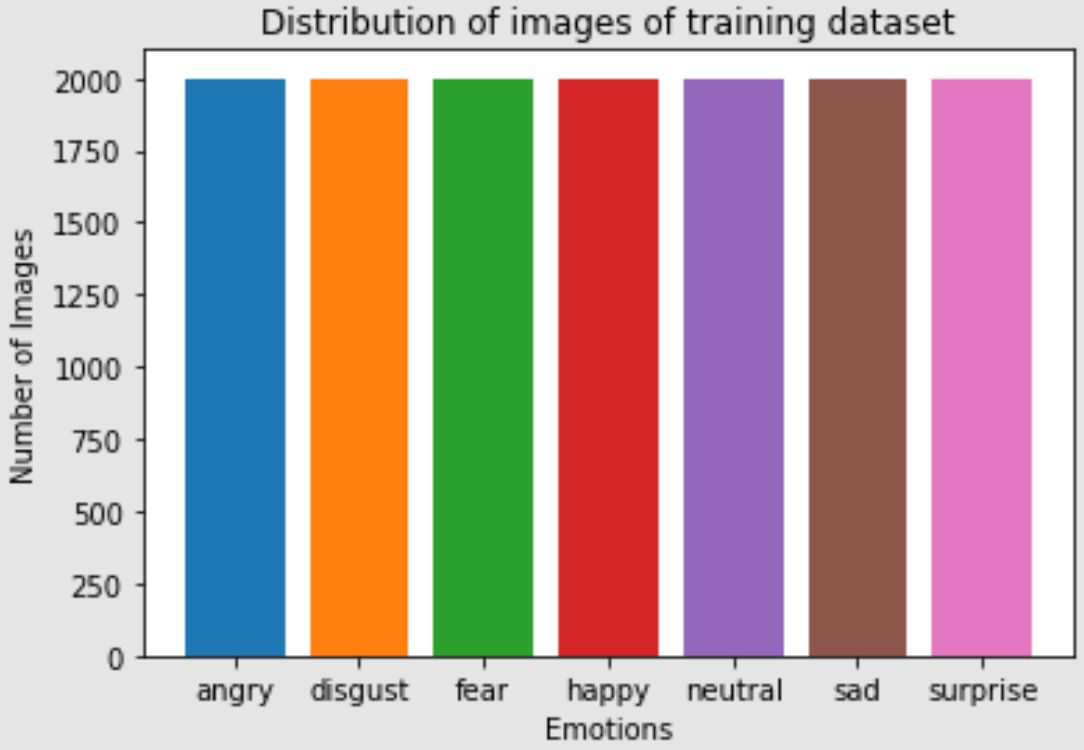
❑ The smallest class is disgust (547 images)

❑ The largest class is happy(8989)



Balancing techniques

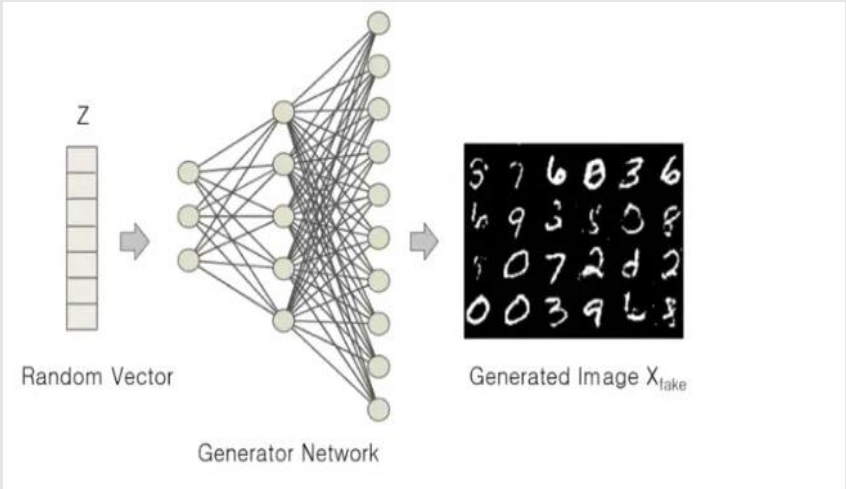
1) GAN
2) SMOTE
3) Upsampling
4) Undersampling
   (The proposed method)

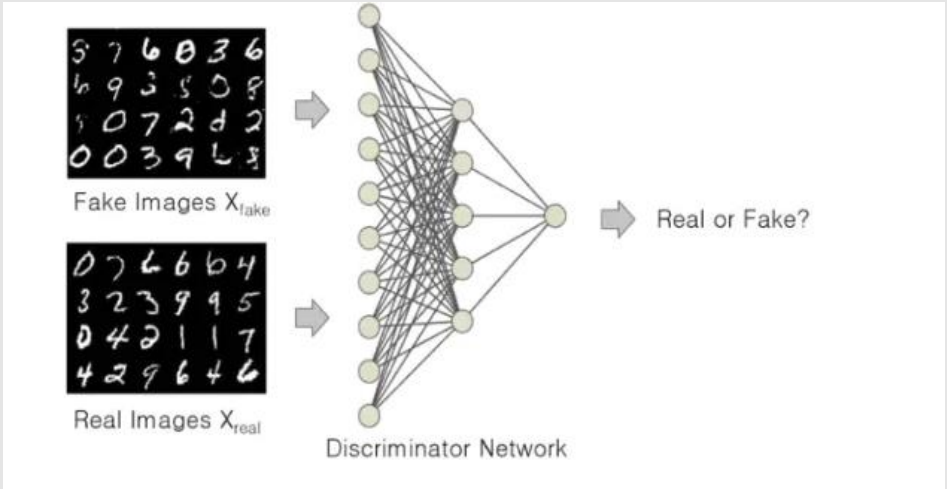


GAN

(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.





GAN

(Generative Adversarial Network)

GENERATOR

The goal of the generator is to *generate* or produce new and synthetic samples of a certain input

Considering the fact that it is ‘competing’ with the discriminator, the generator produce a new fake image with the hope that the discriminator would consider the image to be authentic.

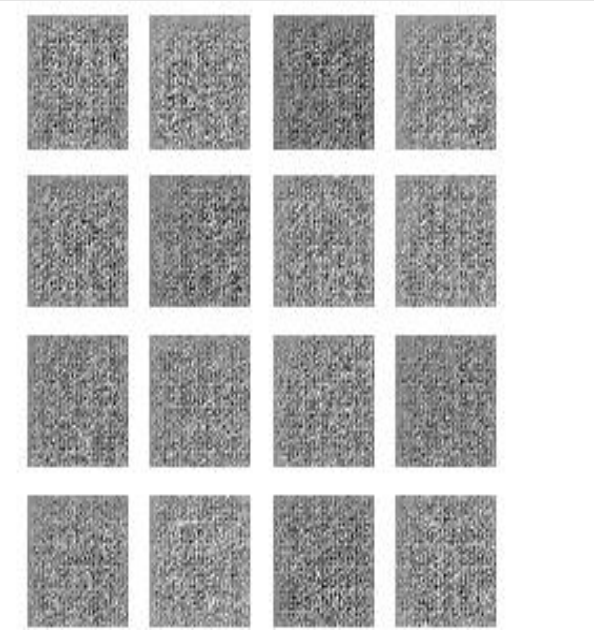
DISCRIMINATOR

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.

tell how "realistic" the input seems, which itself is also being updated dynamically

- 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.

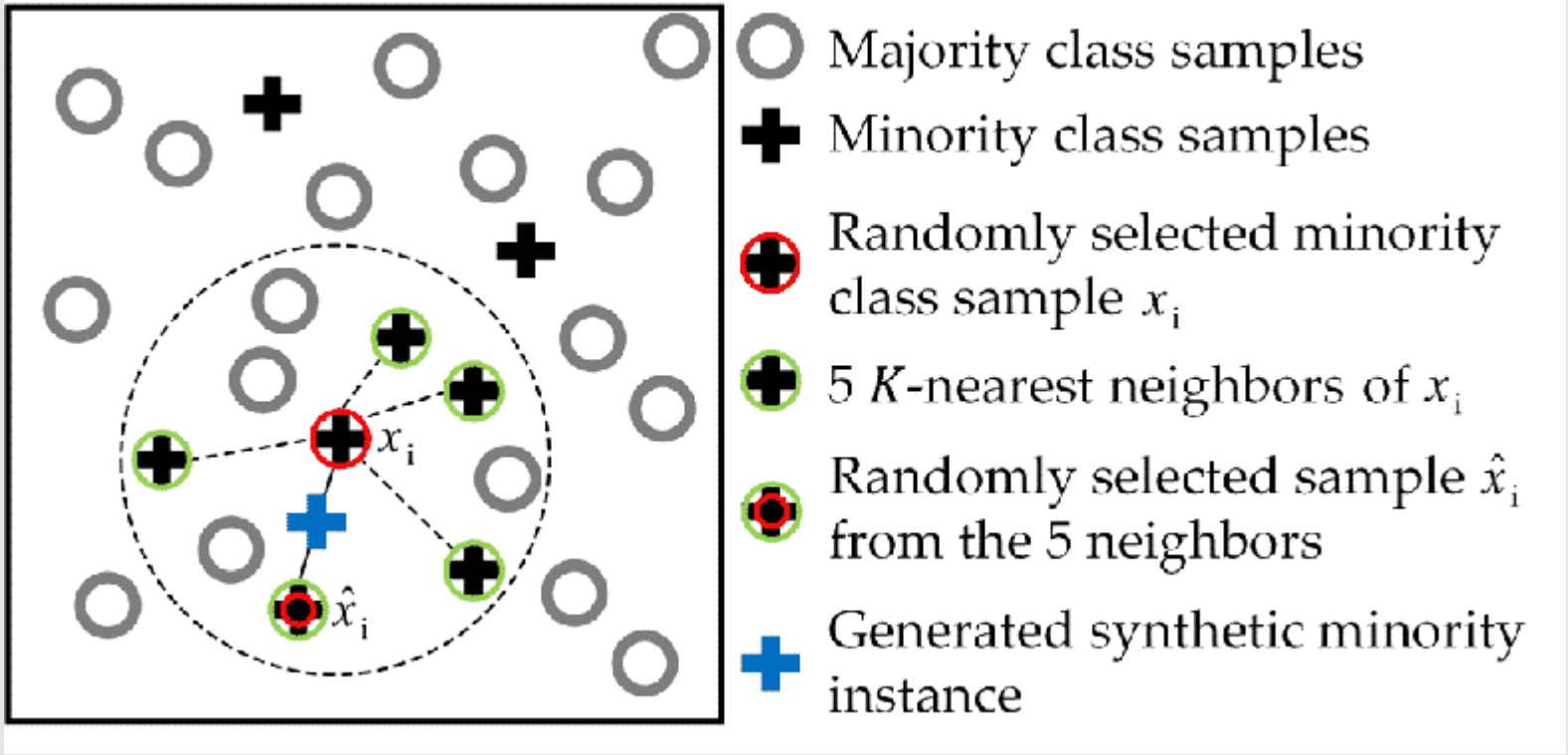


SMOTE

(Synthetic Minority Over-sampling Technique)

- works by selecting a minority class instance and its k nearest neighbors in the feature space.
- It then generates new synthetic instances along the line segments connecting the selected instance and its neighbors.
- The synthetic instances are created by interpolating the feature values between the selected instance and its neighbors.

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 .


x_i
$\hat{x}_i$
Majority class samples
Minority class samples
Randomly selected minority class sample x_i
5 K-nearest neighbors of x_i
Randomly selected sample $\hat{x}_i$ from the 5 neighbors
Generated synthetic minority instance

- Under-sampling

- Up-sampling

- use up-sampling by augmented the small classes and down-sampling the large classes it also gives lo accuracy
- So, we down-sampling all classes to the lowest class disgust(**547**)

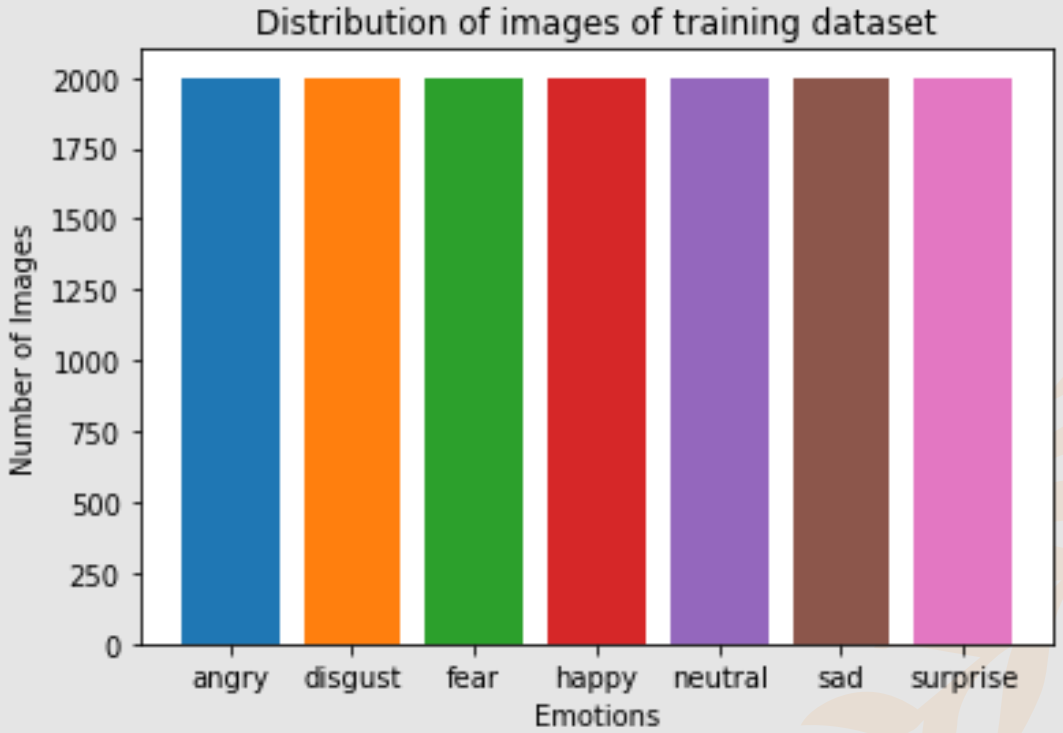
Under-sampling & Up-sampling

Dataset Augmentation

1) intensity variation (-1:2)
(2) flipping (-1:1).

augmentation up-sampling the dataset to 2500 image in each class

Distribution of Balancing Data



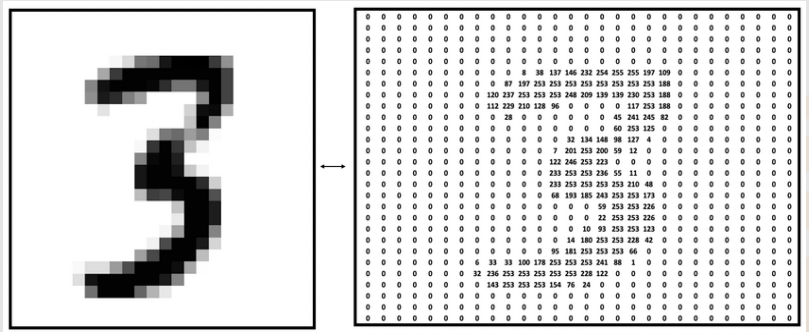
Preparing data for model after balancing

1.Data splitting

- Split the data into train(80%):**2000**
- test(10%):**250**
- validation(10%):**250**

2. Image to array conversion

process of converting image data into a numerical format

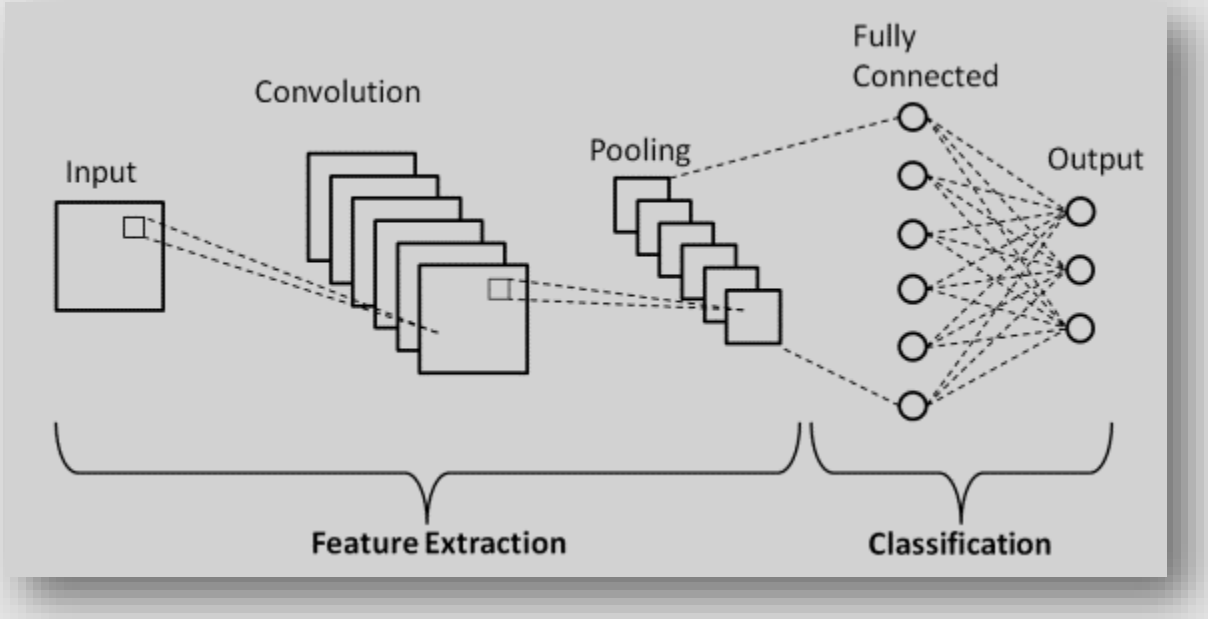


3. Normalization

The process of transforming the columns in a dataset to the same scale **[0,1].**

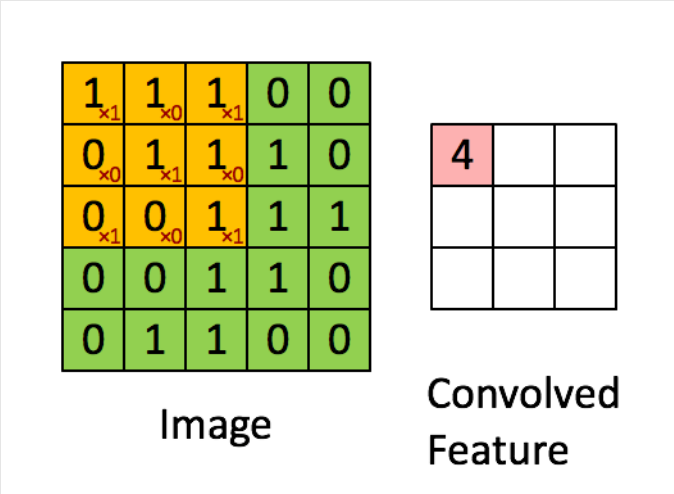
08
ALGORITHMS

CNN model


Fully
Connected
Convolution
Pooling
Input
Output
Feature Extraction
Classification

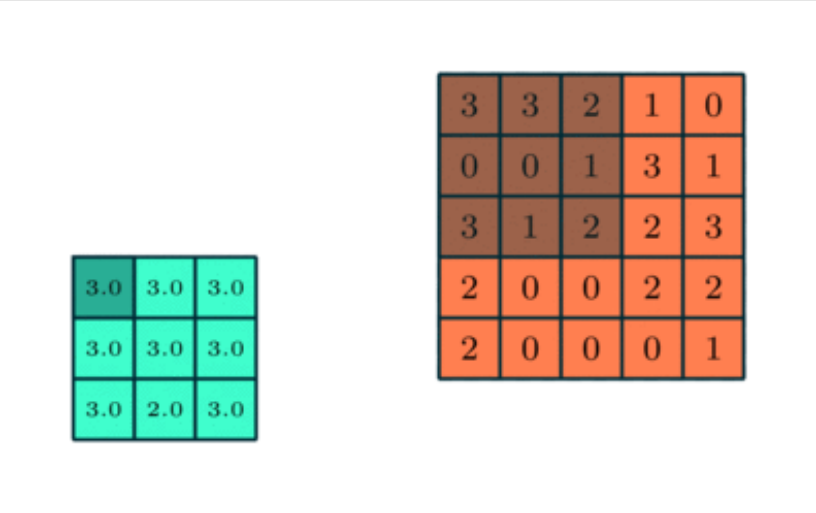
Convolution Layer（ConvNets）

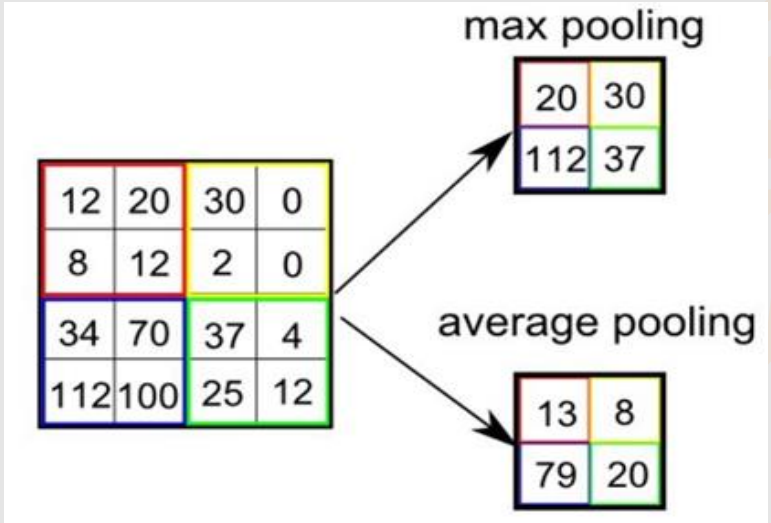
- used to extract high-level features from images, such as edges
- by applying a filter to the input image, with a certain stride length.
- input image is convoluted with the application of filters in CNNs, resulting in a Feature map



Pooling Layer

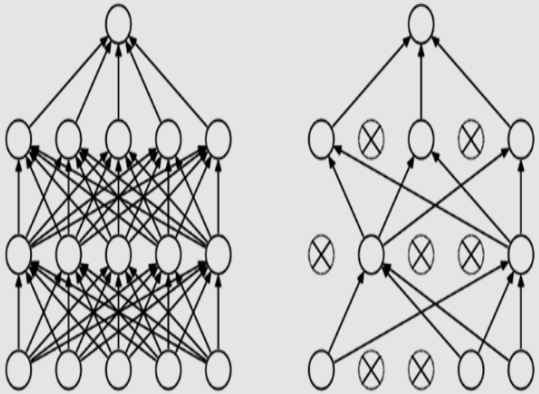
- reduce the spatial size of the Convolved Feature
- decrease the computational power required to process the data
- extract dominant features which are invariant to rotation and position.





Dropout Layer

disregarding certain nodes in a layer at random during training. that prevents overfitting



(a) Standard Neural Net (b) After applying dropout.

Flatten layer

converting the data into a 1-dimensional array

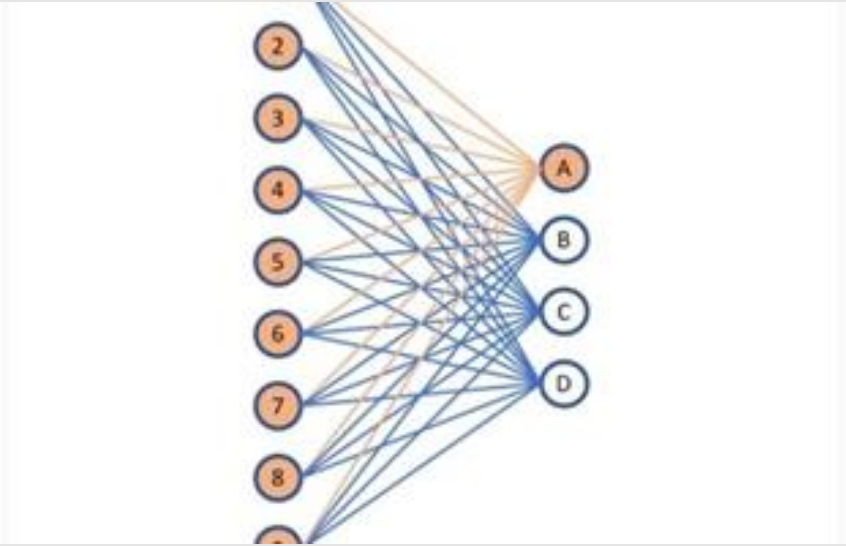
1	1	0
4	2	1
0	2	1

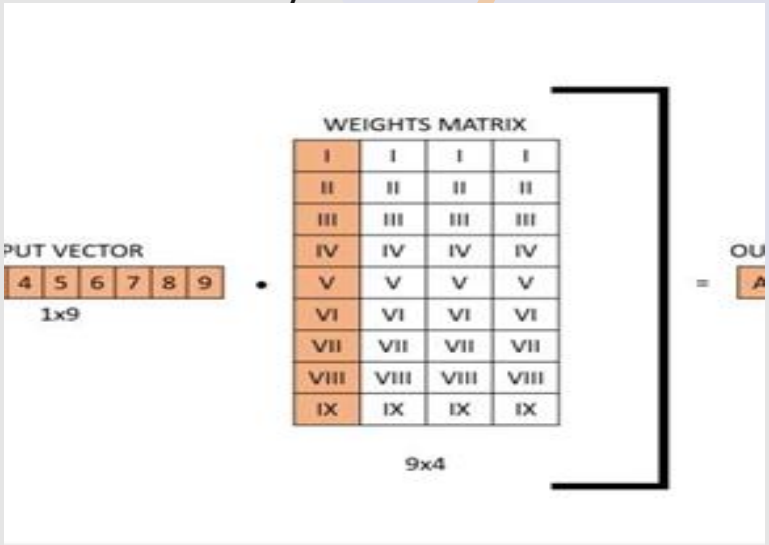
Pooled Feature Map

1
1
0
4
2
1
0
2
1

Fully connected layer

- to learn patterns in the data provided.
- by using weights to map the inputs from the previous layer to the next layer.
- Each neuron in the layer is connected to all the neurons in the previous layer, thus the name “fully connected”.





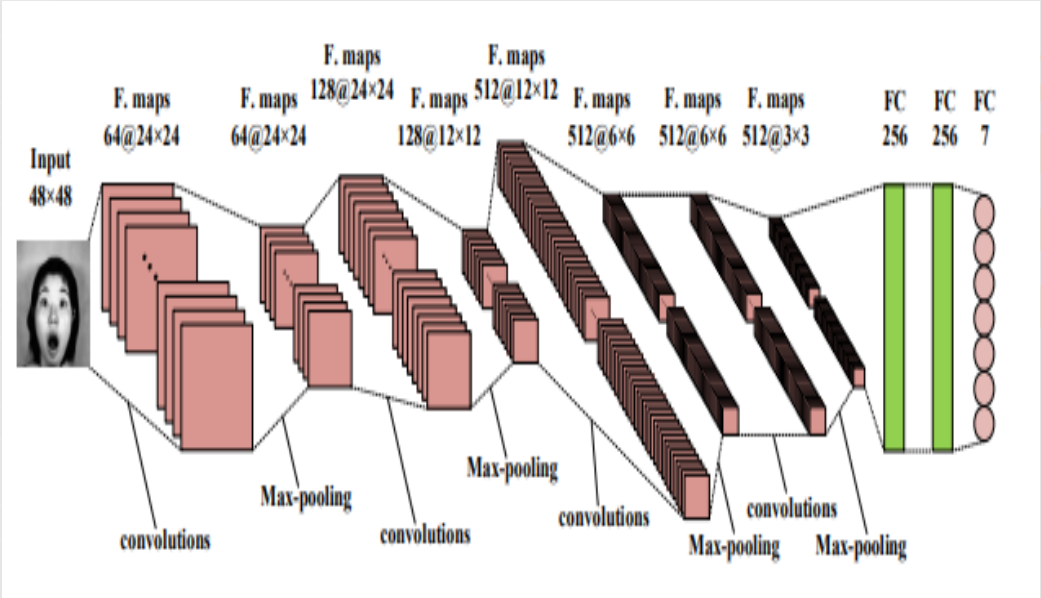
output Layer(Softmax)

- **The Softmax function takes in the values from the previous layer**
- results in a probability distribution, which the neural network can use to make predictions.

Class	Probability
apple	0.001
bear	0.04
candy	0.008
dog	0.95
egg	0.001

CNN structure

- 4 convolution layers (3,3) kernel size
- **Dense layers:**
  - 2 dense layers **(output)**
  - Optimizer
  - **ADAM**
  - Activation function (RELU)
- Max pooling (2,2)

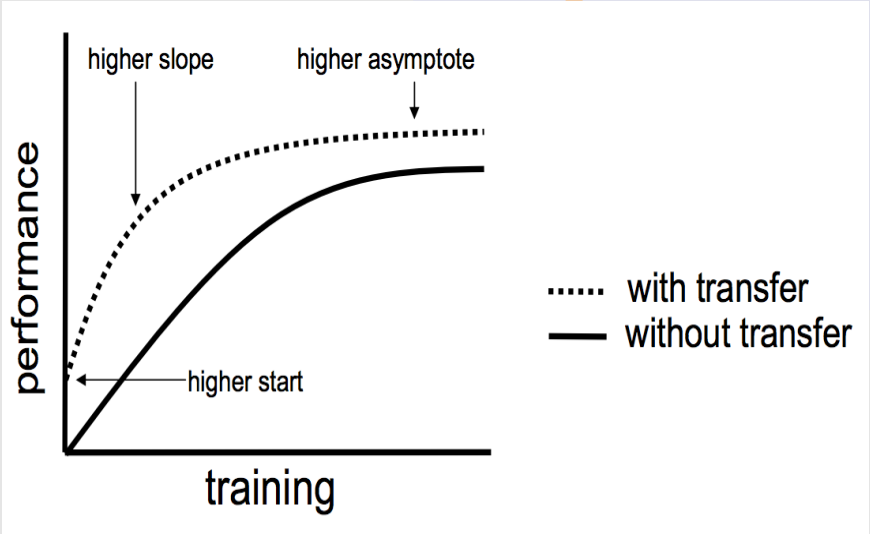


Transfer learning

Transfer learning idea

technique in deep learning that allows us to leverage pre-trained models and their learned features to achieve better performance and faster convergence

Instead of training a model from scratch, transfer learning enables us to use a pre-trained model as a starting point and fine-tune it for our specific task.



Why we use transfer learning?

include faster training time,

improved accuracy

reduced cost of training a model from scratch

valuable tool for solving tasks where data is limited.

Pretrained models

VGG16

ResNet50

DenseNet201

MobileNet

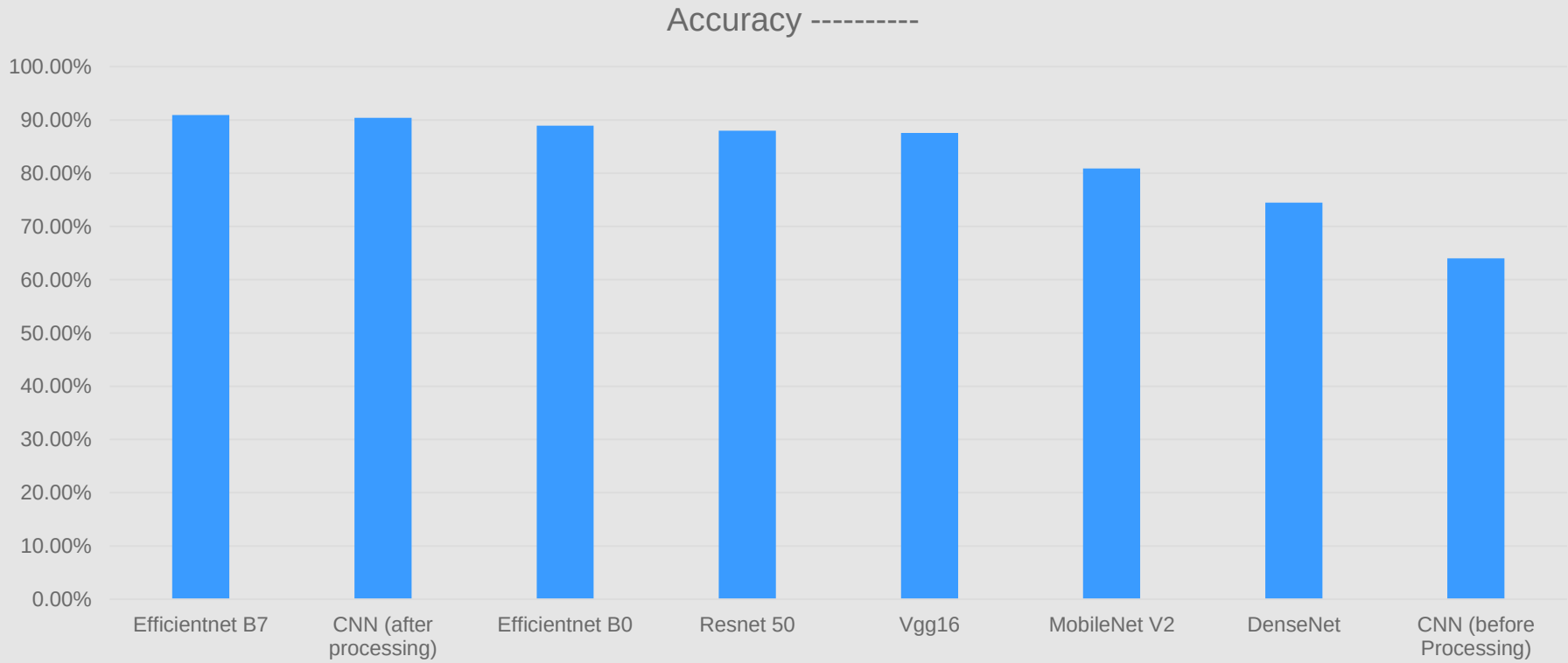
EfficientNetB0

EfficientNetB7

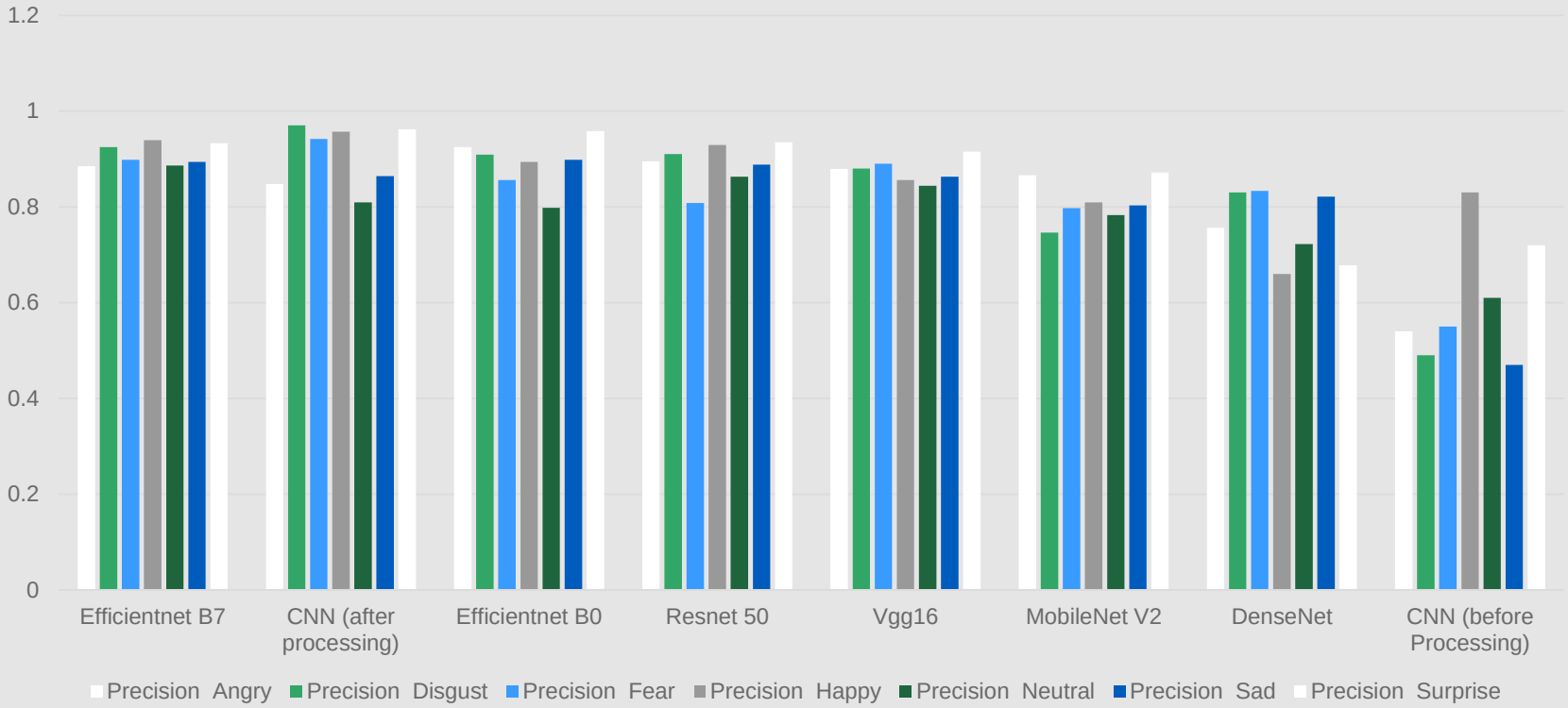
09
RESULTS
&
DISCUSSION

Classification report	Facial Expression	Efficientnet B7	CNN (after processing)	Efficientnet B0	Resnet 50	Vgg16	MobileNet V2	DenseNet	CNN (before Processing)
Accuracy	----------	**90.91%**	**90.40%**	**88.91%**	**88%**	**87.54%**	**80.86%**	**74.46%**	**64%**
Epochs	----------	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**80**
Precision	Angry	0.885	0.848	0.925	0.895	0.879	0.866	0.756	0.54
	Disgust	0.925	0.970	0.909	0.91	0.880	0.746	0.83	0.49
	Fear	0.898	0.942	0.856	0.808	0.890	0.797	0.833	0.55
	Happy	0.939	0.957	0.894	0.9294	0.856	0.809	0.66	0.83
	Neutral	0.886	0.809	0.798	0.8629	0.844	0.783	0.722	0.61
	Sad	0.894	0.864	0.898	0.8884	0.863	0.803	0.821	0.47
	Surprise	0.933	0.962	0.958	0.9349	0.915	0.872	0.678	0.72
Recall	Angry	0.864	0.896	0.84	0.860	0.848	0.776	0.82	0.54
	Disgust	0.944	0.932	0.924	0.932	0.94	0.872	0.74	0.66
	Fear	0.884	0.852	0.856	0.896	0.848	0.788	0.58	0.43
	Happy	0.936	0.900	0.912	0.896	0.88	0.796	0.8	0.84
	Neutral	0.908	0.920	0.888	0.856	0.844	0.796	0.768	0.53
	Sad	0.880	0.892	0.884	0.860	0.856	0.784	0.644	0.57
	Surprise	0.948	0.936	0.92	0.920	0.912	0.848	0.86	0.80
F-Score	Angry	0.874	0.871	0.8805	0.8775	0.863	0.818	0.787	0.54
	Disgust	0.934	0.951	0.9166	0.9209	0.909	0.804	0.782	0.56
	Fear	0.891	0.895	0.856	0.8501	0.868	0.793	0.684	0.48
	Happy	0.9378	0.928	0.903	0.9124	0.867	0.802	0.723	0.83
	Neutral	0.897	0.861	0.8409	0.8595	0.844	0.789	0.744	0.56
	Sad	0.887	0.878	0.891	0.8739	0.859	0.793	0.721	0.52
	Surprise	0.940	0.95	0.9387	0.9274	0.913	0.860	0.758	0.76

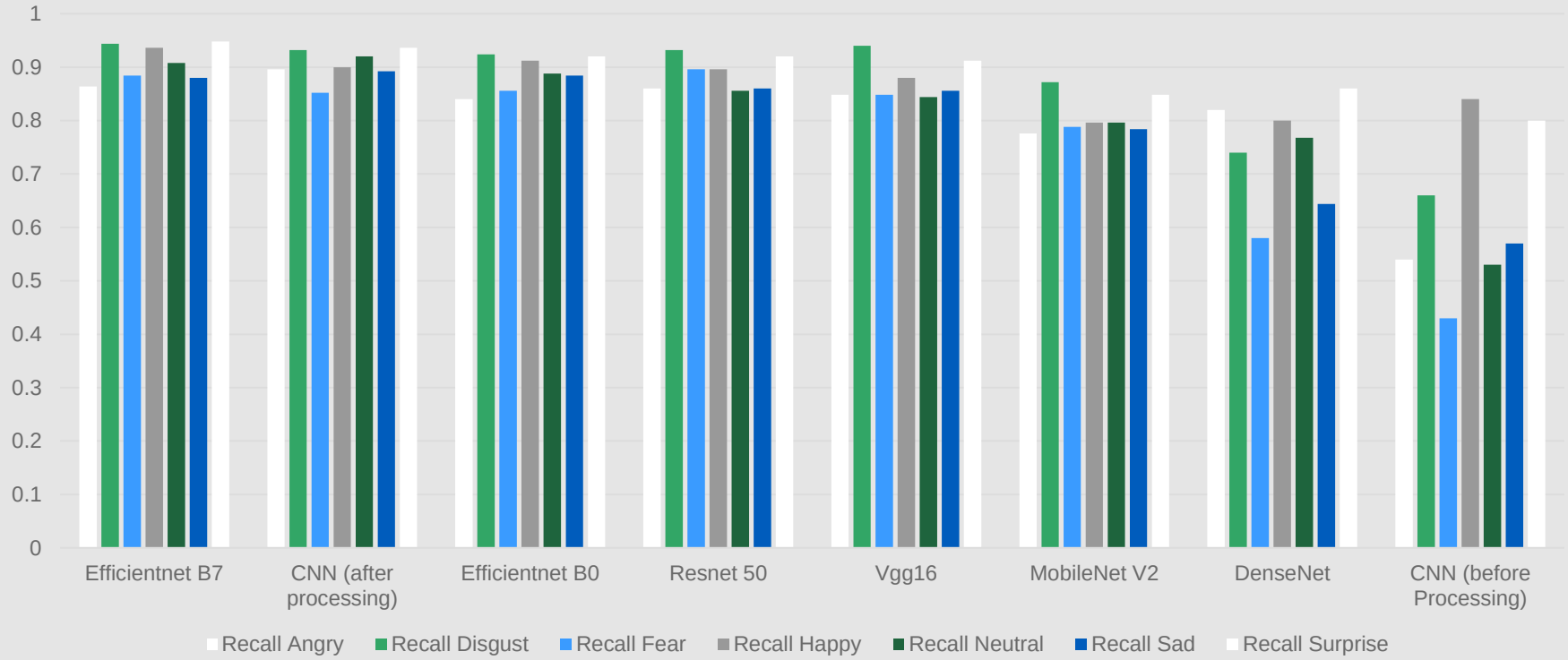
Accuracy Chart



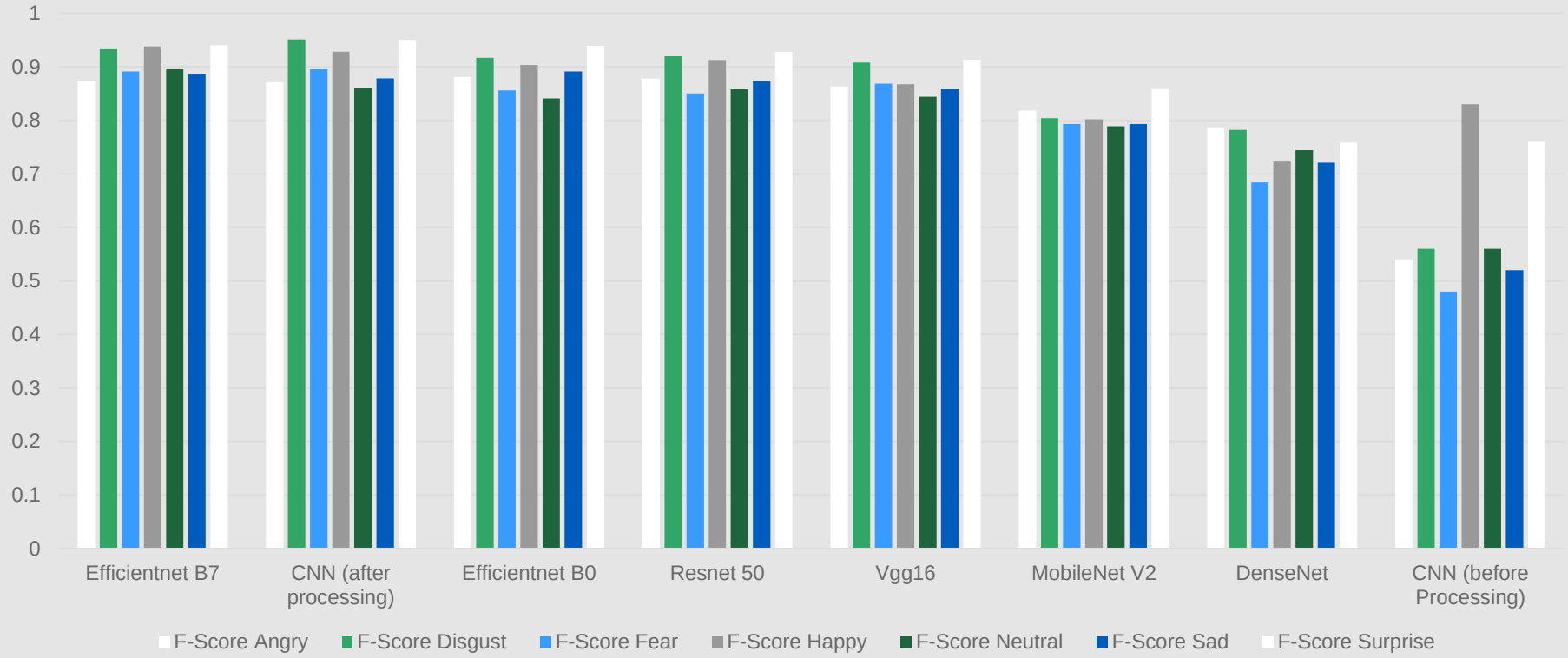
Precision chart



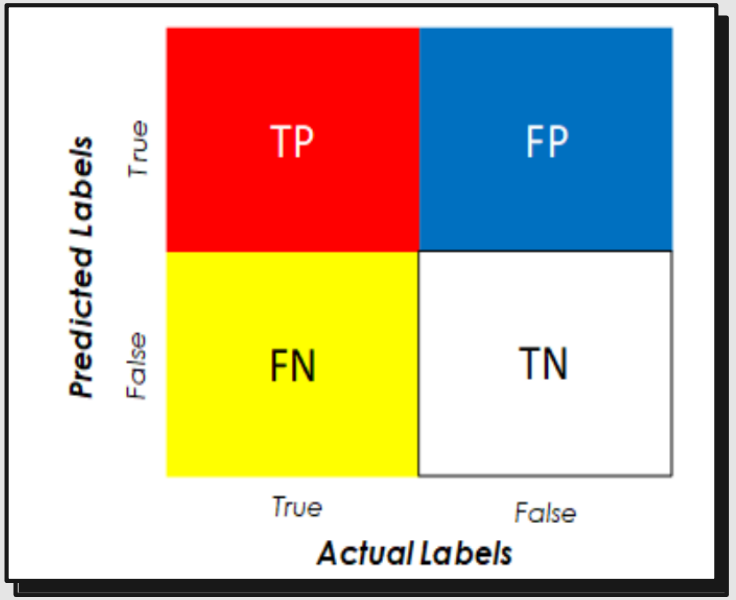
Recall chart



F-Score chart

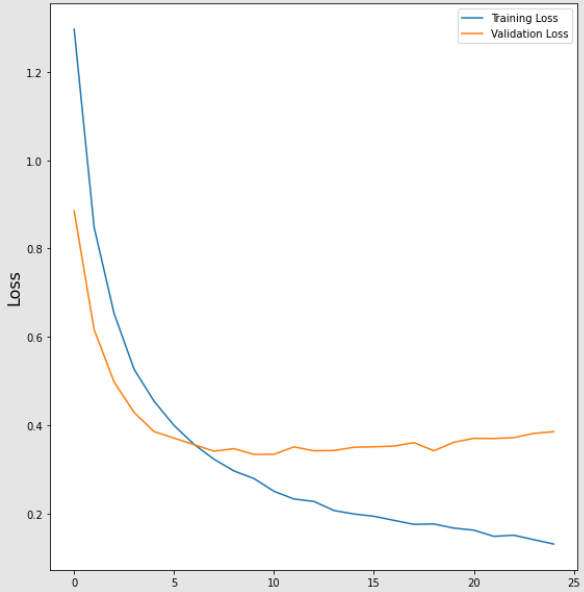


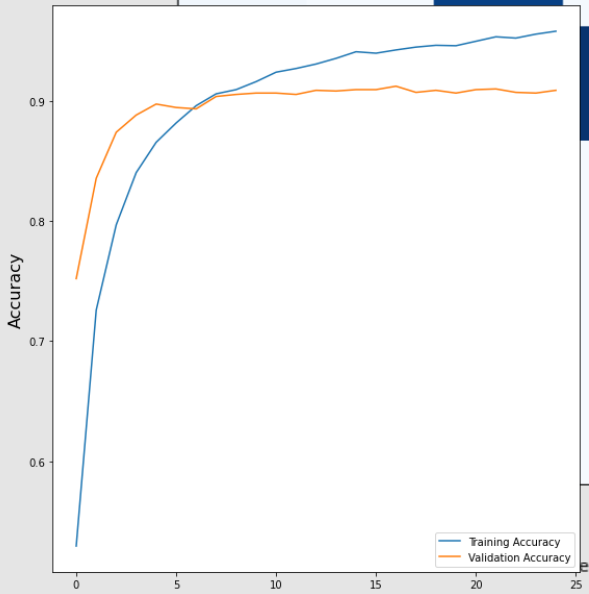
CONFUSION MATRIX



EfficientNetB7

Optimizer : Adam

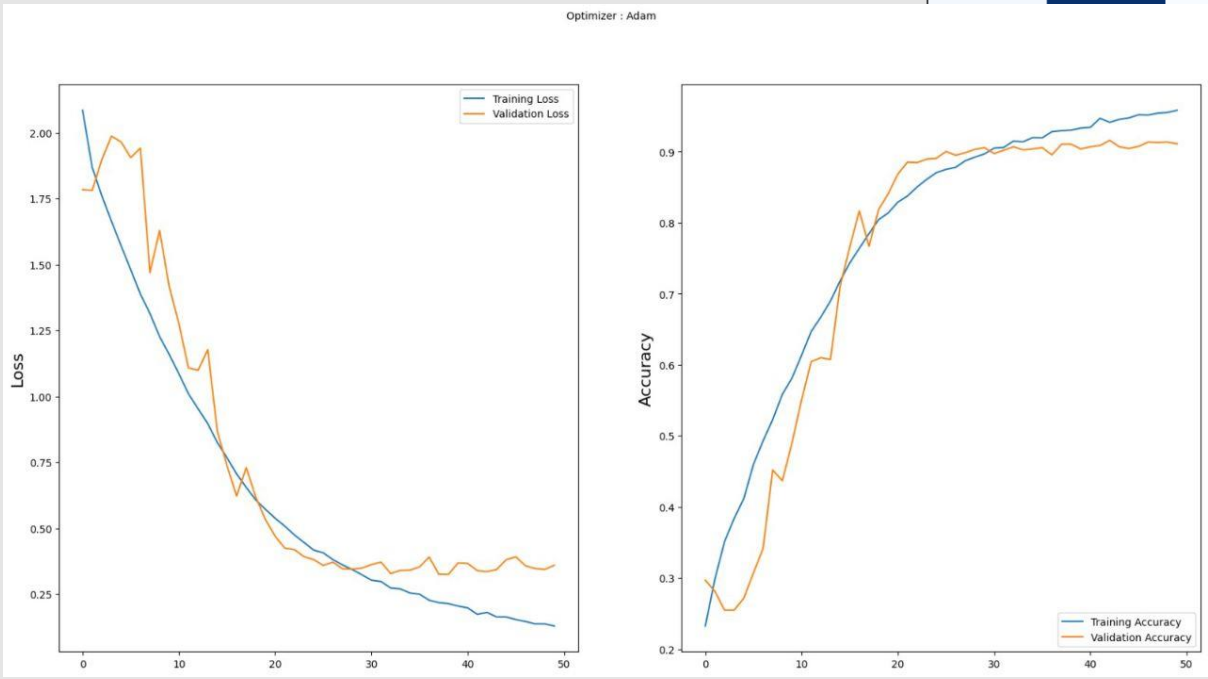


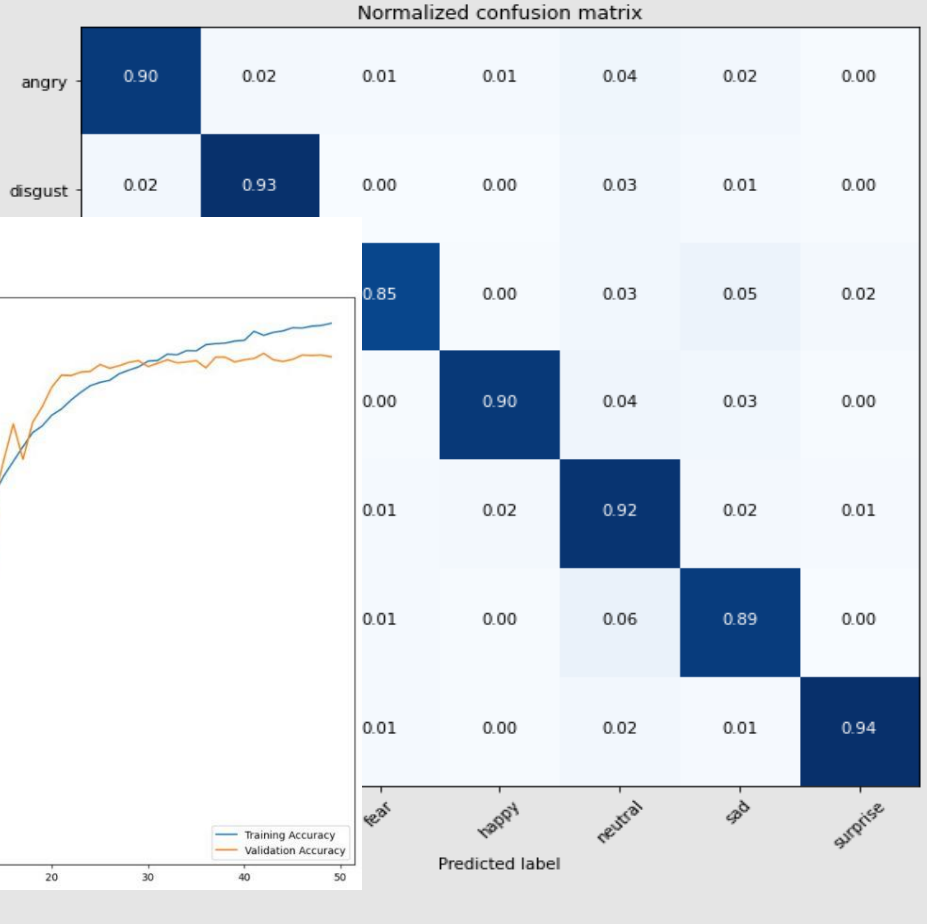


Prediction

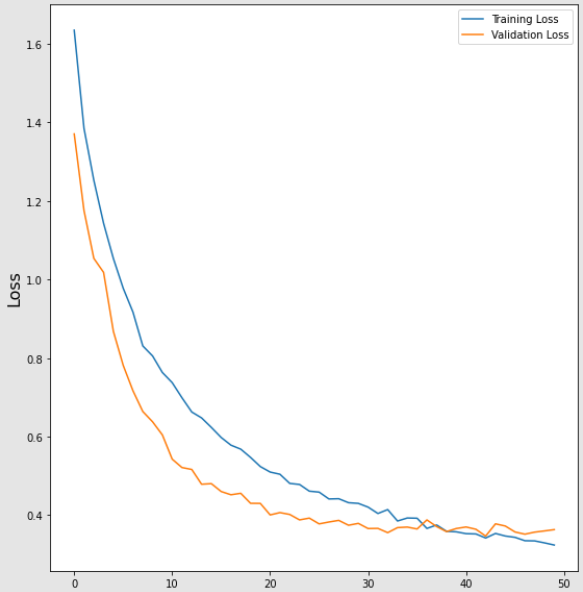

Actual: fear
Predicted: fear
Actual: sad
Predicted: fear
Actual: neutral
Predicted: neutral
Actual: fear
Predicted: fear
Actual: fear
Predicted: fear
Actual: happy
Predicted: happy
Actual: neutral
Predicted: neutral
Actual: sad
Predicted: sad
Actual: happy
Predicted: happy
Actual: neutral
Predicted: neutral
Actual: fear
Predicted: fear
Actual: happy
Predicted: happy
Actual: disgust
Predicted: disgust
Actual: happy
Predicted: happy
Actual: fear
Predicted: fear
Actual: fear
Predicted: fear

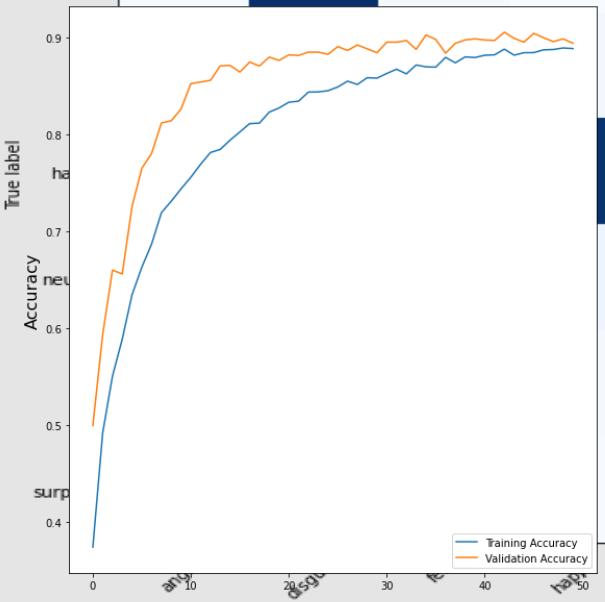
CNN(after processing)



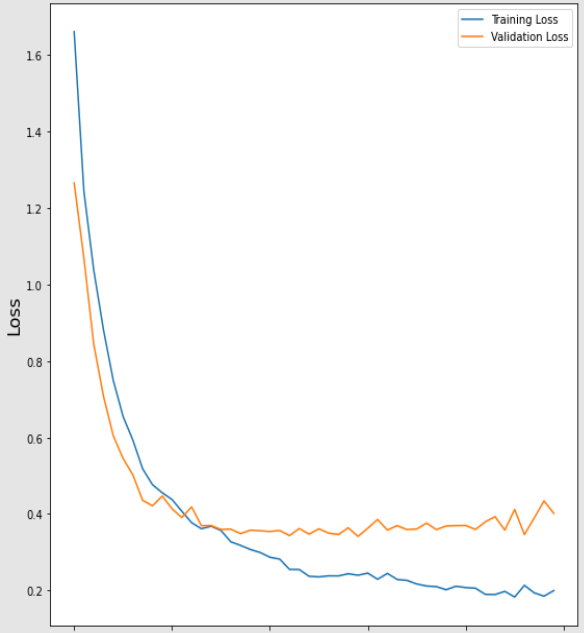


EfficientNetB0

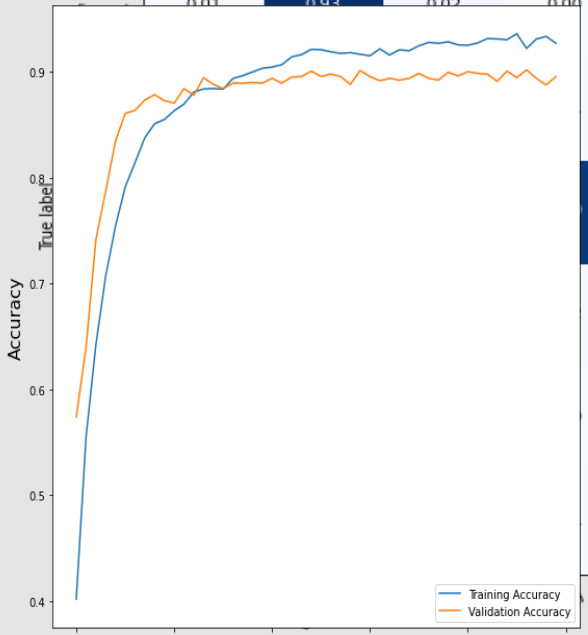




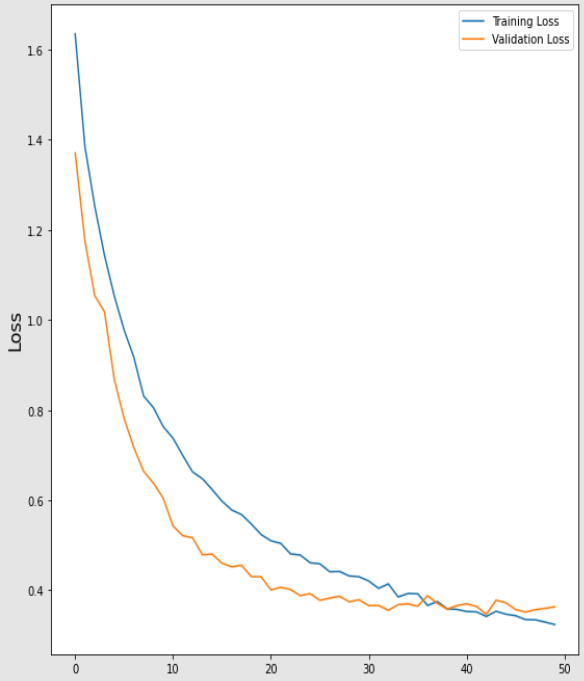
Resnet 50



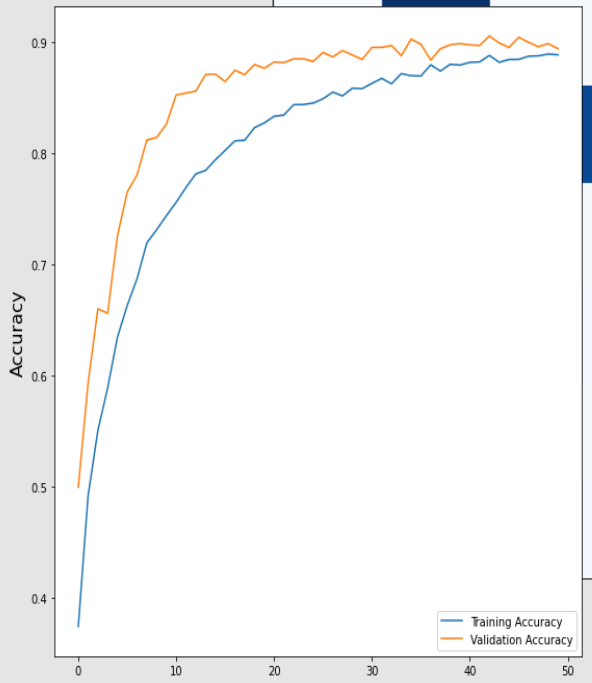
Optimizer : Adam



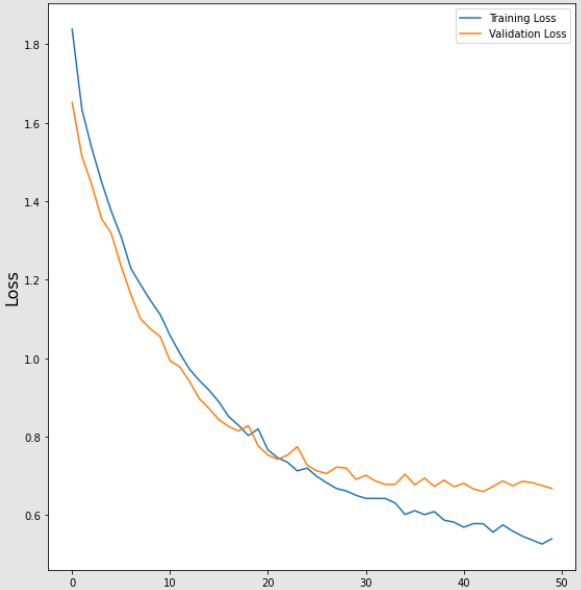
Vgg 16



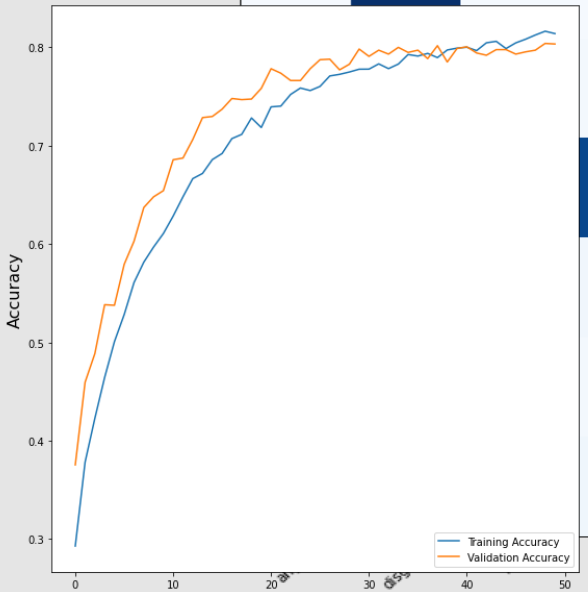
Optimizer : Adam



MobileNetV2

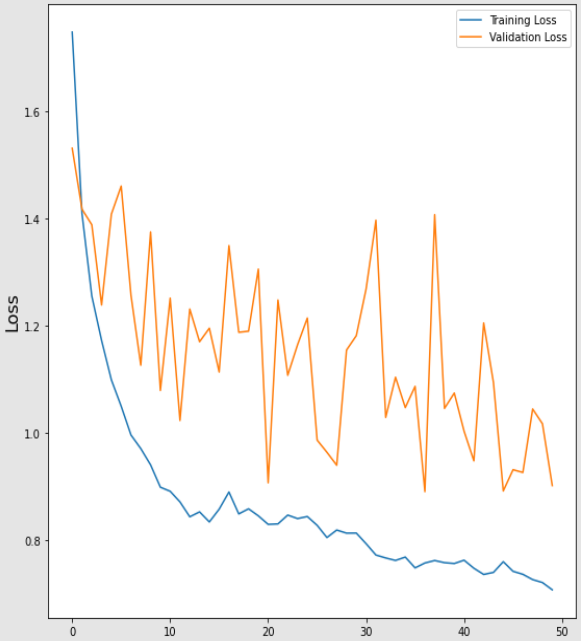


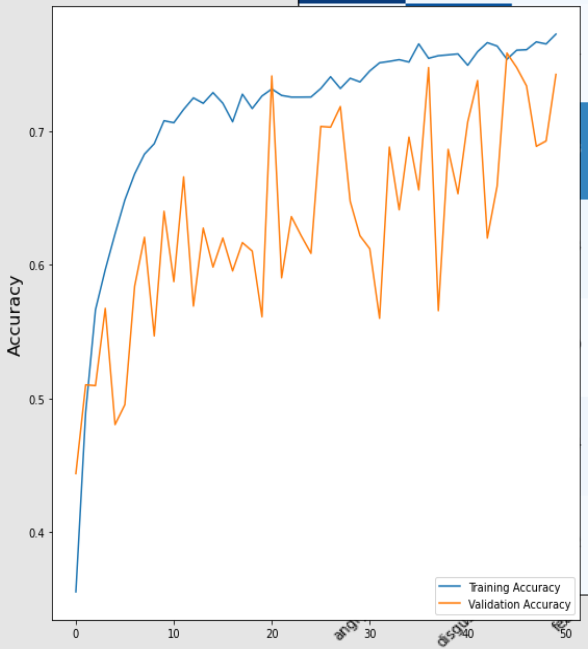
Optimizer : Adam



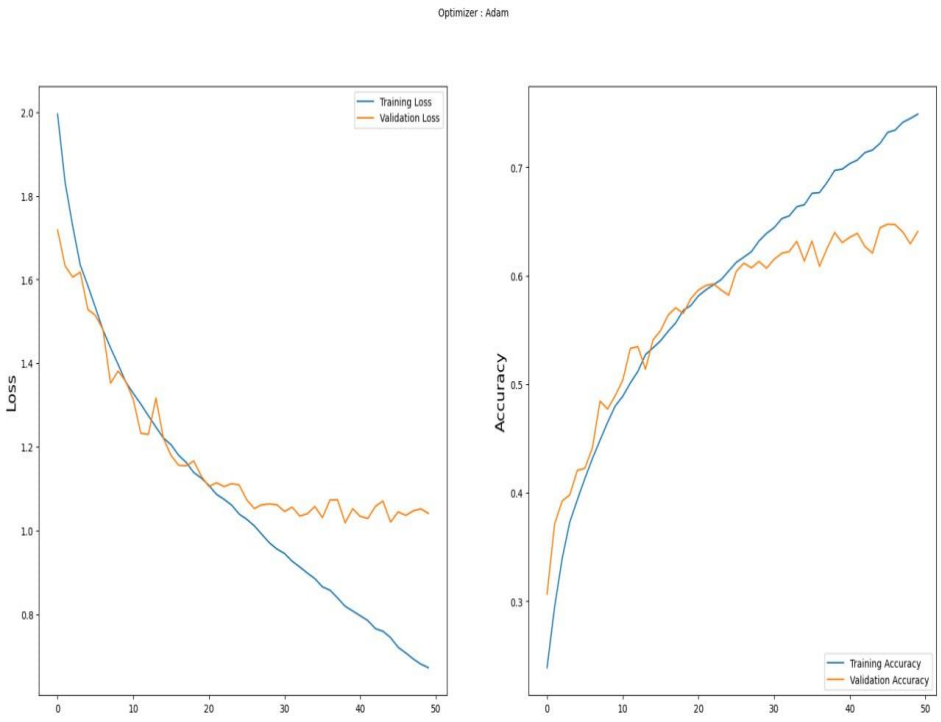
DenseNet

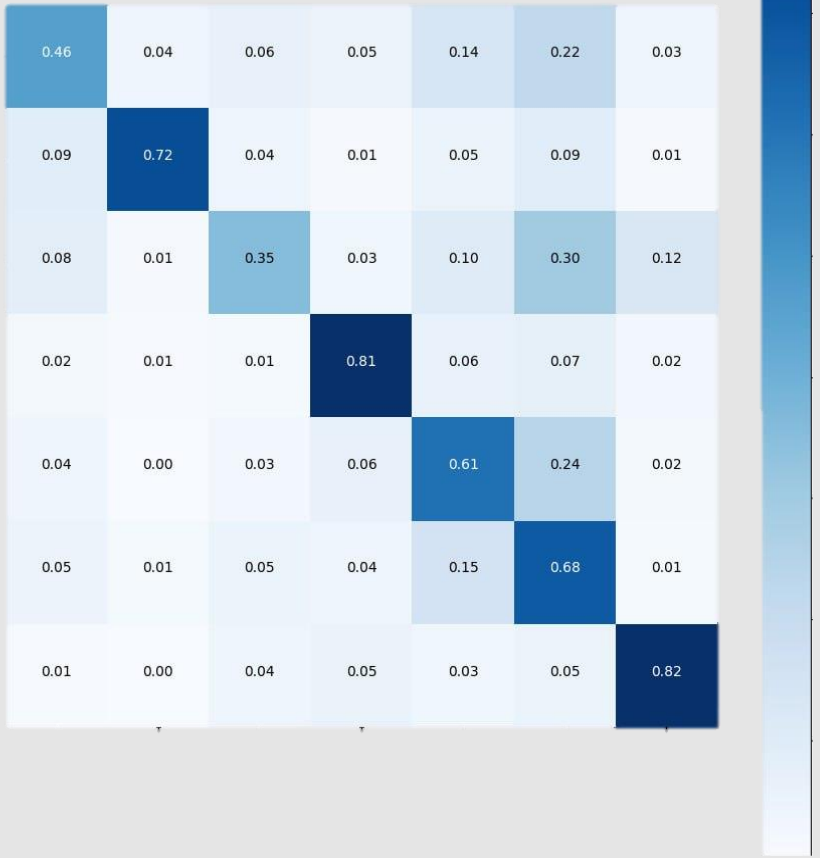
Optimizer : Adam





CNN (before Processing)





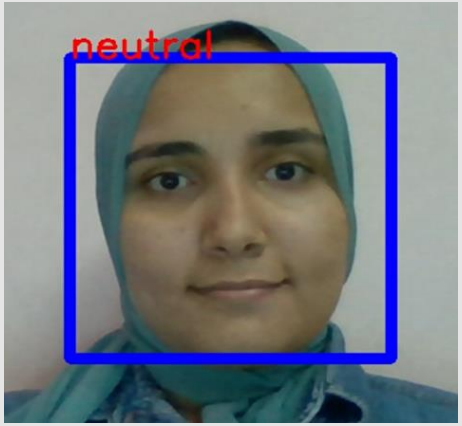
10

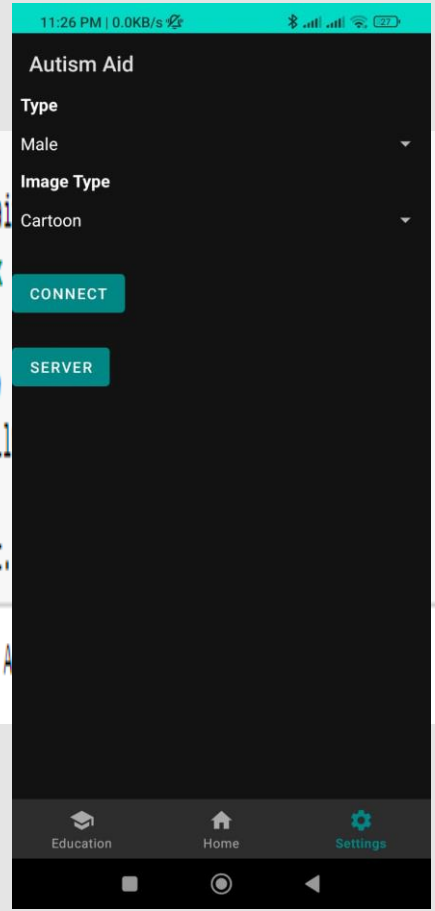
MODEL INTERFACING WITH MOBILE APPLICATION

1) Model with Laptop Camera

- Saving the weights by **model.save** that save the weights in form of .h5 file by using also The ModelCheckpoint callback the main purpose of it is save the model's weights or entire model to disk at specific points during training.
- Then load the model by using OpenCV camera library and Tensorflow and keras libraries to load the weights of the model and open the camera of laptop for prediction.

Prediction using laptop camera





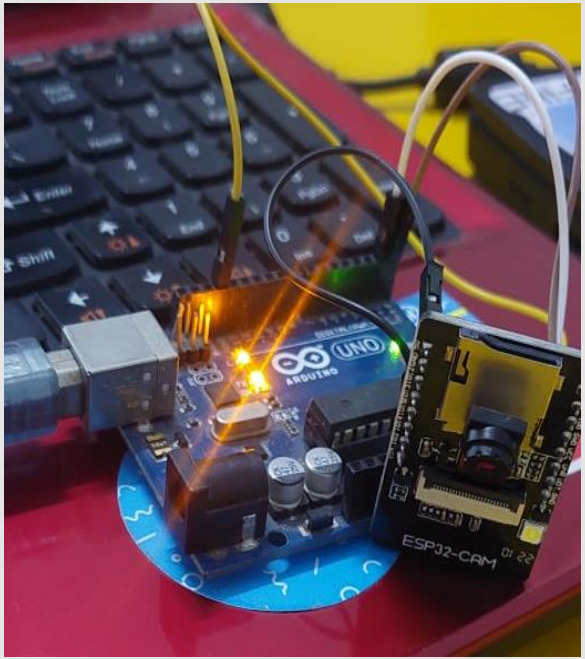
2) Mobile application with laptop camera

- Connecting the mobile application with the model Via Bluetooth.
- After choosing the application port and the Bluetooth address of the mobile, write it in python code then click on server button in mobile application settings to connect the camera and send predictions to mobile application

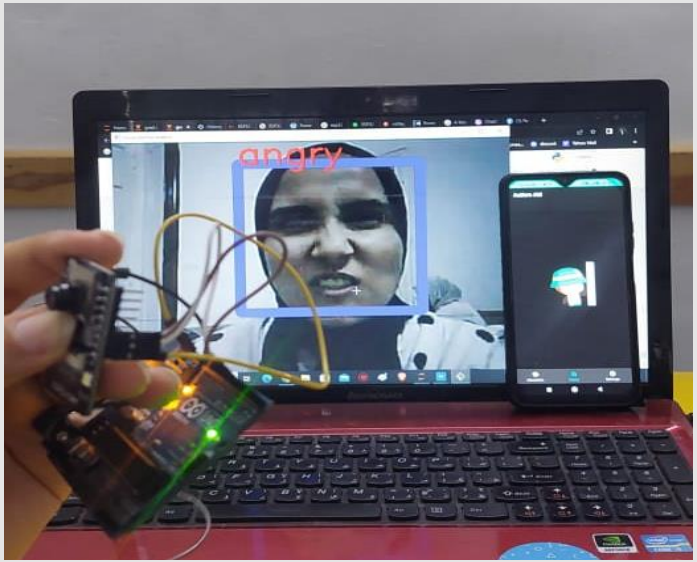

happy

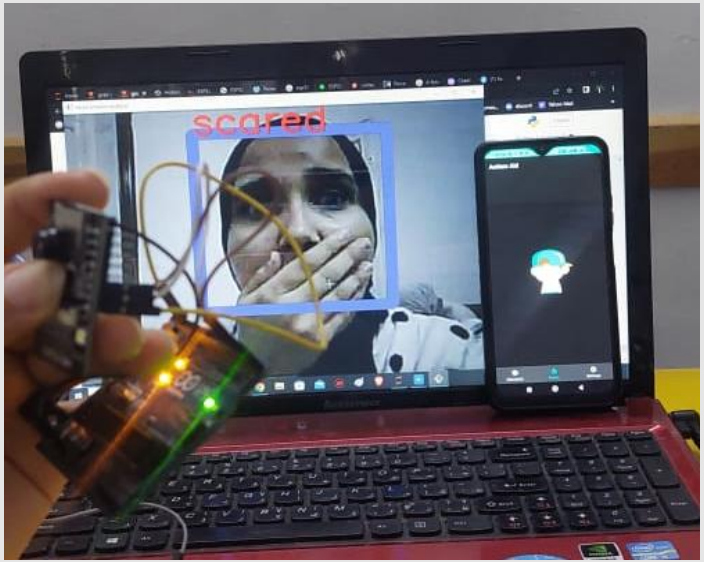
3)Hardware

- Using Esp32 camera instead of laptop camera to implement our project in the real world.
- Fist of all we use Arduino uno to communicate with Esp32 camera.
- Connecting the Esp32 camera to the same network of the laptop to provide the access .
- Esp32 camera capturing the images and laptop send predictions to mobile application.



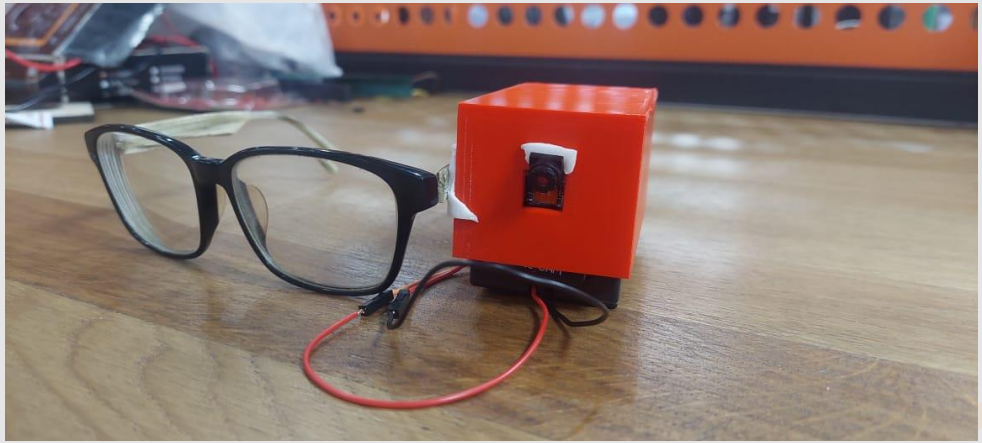
Esp32 camera prediction





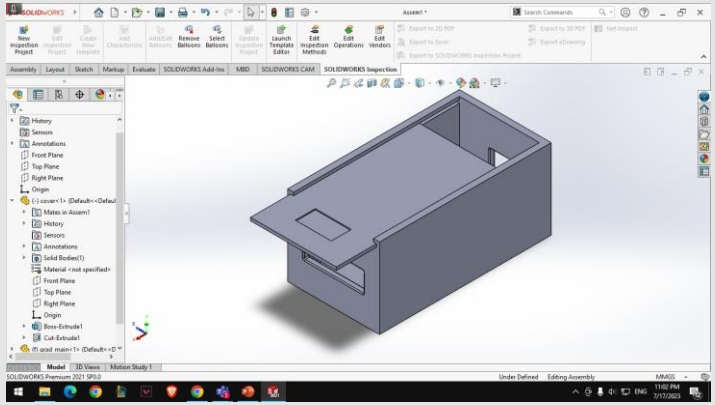
3) Hardware

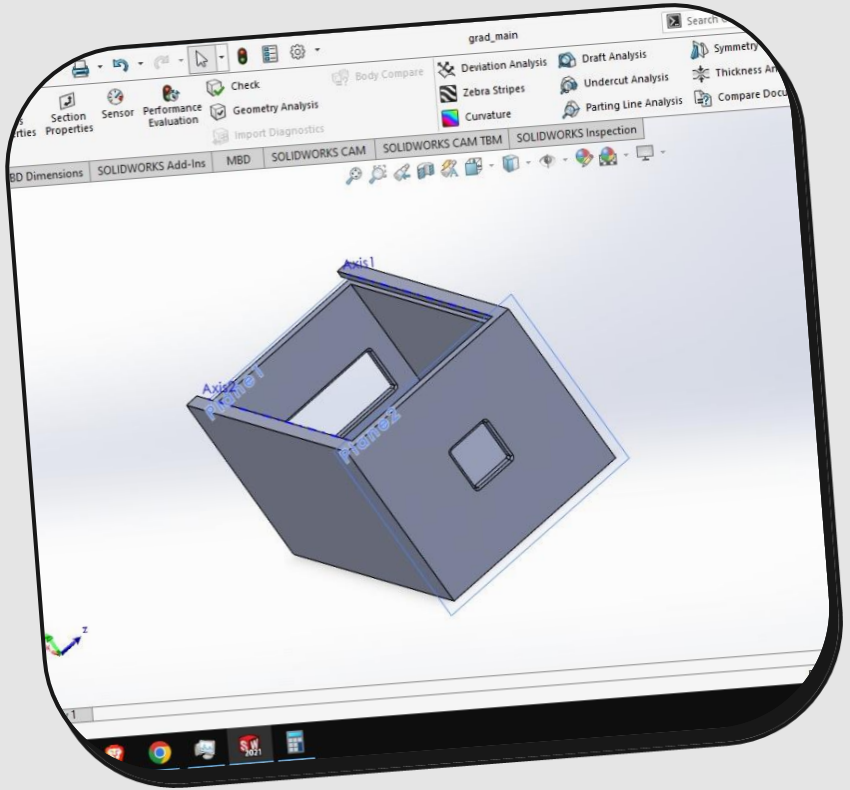
Powering the Esp32 camera by using 2 lithium-ion batteries with their holder and type c charger module for lithium-ion batteries charging, then casing all components by making a 3d printing box with a specific dimension.



3)Hardware

- Using solid-works for designing the casing the hardware components





11

Problems

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, w 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.

Problems and solutions

Problem	Solutions
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.
Try to use dc battery 9v but it didn’t work (to optimize the size of casing)	But not compatible with the camera so we used the lithium-ion batteries

12
Cost

Hardware

Esp32 Camera

450 L.E

Arduino Uno

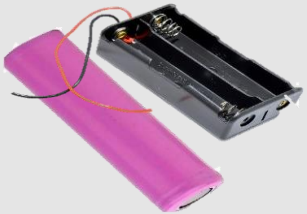
365 L.E

Type C module

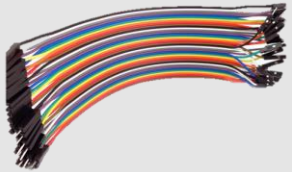
35 L.E

Lithium-ion batteries and holder

110 L.E



Jumpers

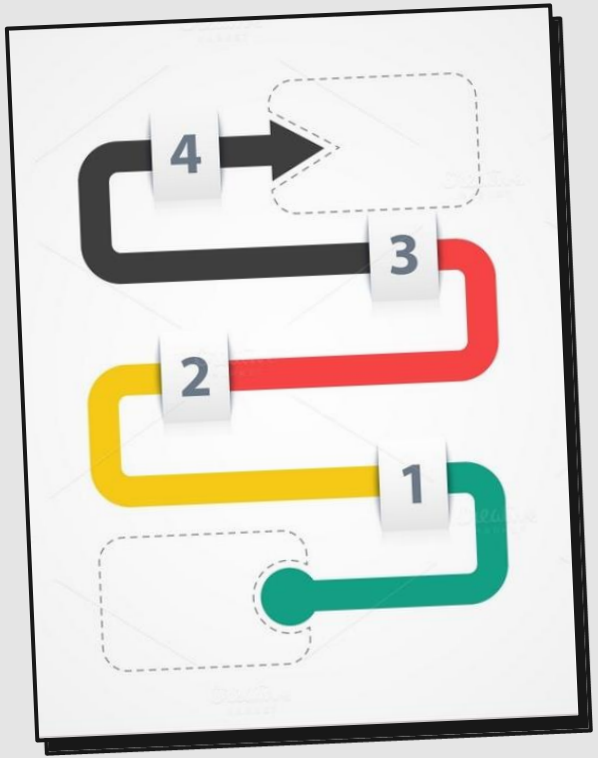


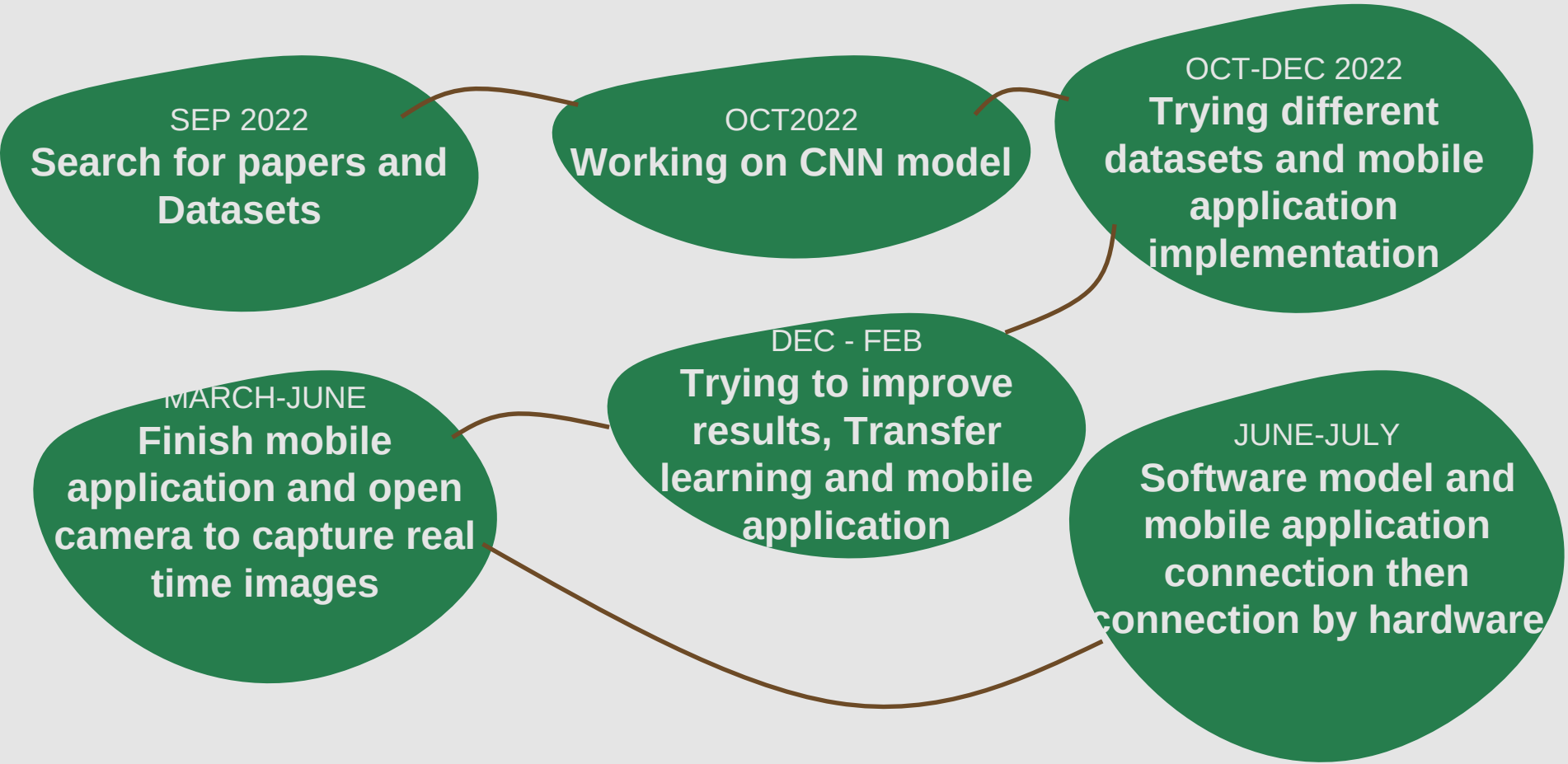
Casing

120 L.E

Total cost:1095 L.E

13 Timeline




SEP 2022
Search for papers and Datasets
OCT2022
Working on CNN model
OCT-DEC 2022
Trying different datasets and mobile application implementation
DEC - FEB
Trying to improve results, Transfer learning and mobile application
MARCH-JUNE
Finish mobile application and open camera to capture real time images
JUNE-JULY
Software model and mobile application connection then connection by hardware

Thank you