A

Major Project On

Automated Engagement Recognition in E-Environments

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

S. Uday Sai (187R1A0552)

Ankita Mishra (187R1A0509)

Under the Guidance of

J. NARASIMHARAO

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**AUTOMATED ENGAGEMENT RECOGNITION IN E-ENVIRONMENTS**" being submitted by **S. UDAY SAI (187R1A0552)** and **ANKITA MISHRA (187R1A0509)** in partial fulfillment of the requirements for the award of the degree of B. Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mr. J. Narasimharao
Associate Professor
INTERNAL GUIDE

Dr. A. RajiReddy
DIRECTOR

Dr. K. Srujan Raju
HoD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on_____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. J. Narasimharao,** Associate Professor, for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Mr. J. Narasimharao, Dr. T. S. Mastan Rao, Dr. Suwarna Gothane, Mr. A. Uday Kiran, Mrs. G. Latha, Mr. A. Kiran Kumar** for their cordial support, valuable information, and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

ANKITA MISHRA (187R1A0509)

S.UDAY SAI (187R1A0552)

ABSTRACT

The gap between the actual and virtual worlds is closing at an incredible rate. Interaction with computers is becoming increasingly common as more people utilize them to complete a variety of jobs ranging from online learning to shopping. In such circumstances, identifying a user's level of involvement with the system with which he or she is engaging can modify how the system responds to the user. This will result in more engagement with the system as well as improved human-computer connection. In today's vision applications, including advertising, healthcare, autonomous vehicles, and e-learning, identifying user engagement might be critical. An automated engagement detection system that can analyze a person’s engagement outcome with a certain object or an environment can be crucial to many organizations and businesses around the globe. Therefore, we employ cutting-edge algorithms in our project to recognize user engagement levels and divide them into two categories: positive and negative.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

This project is titled “AUTOMATED ENGAGEMENT RECOGNITION IN E-ENVIRONMENTS”. An automated engagement detection system that can analyze a person’s engagement outcome with a certain object or an environment in an E-Environment. We built a computer vision model that takes input from a video, recognizes human emotions from face expressions and body behavior and categorizes them into either positive or negative. This can be further developed to track and detect in real time and send the information to the backend for further use cases.

1.2 PROJECT PURPOSE

The gap between the actual and virtual worlds is closing at an incredible rate. Interaction with computers is becoming increasingly common as more people utilize them to complete a variety of jobs ranging from online learning to shopping. In such circumstances, identifying a user's level of involvement with the system with which he or she is engaging can modify how the system responds to the user. This will result in more engagement with the system as well as improved human-computer connection. In today's vision applications, including advertising, healthcare, autonomous vehicles, and e-learning, identifying user engagement might be critical. We automate engagement level recognition for E-Environments using advanced computer vision techniques such as Slow Fast networks.

1.3 PROJECT FEATURES

The main feature of this project is that the system will be capable of identifying the different states of emotions a user goes through in a E-setting and analyze it and categorize whether the response is either positive or negative without

any human intervention.

2. SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

The gap between the actual and virtual worlds is closing at an incredible rate. Interaction with computers is becoming increasingly common as more people utilize them to complete a variety of jobs ranging from online learning to shopping. In such circumstances, identifying a user's level of involvement with the system with which he or she is engaging can modify how the system responds to the user. This will result in more engagement with the system as well as improved human-computer connection. In today's vision applications, including advertising, healthcare, autonomous vehicles, and e-learning, identifying user engagement might be critical. An automated engagement detection system that can analyze a person’s engagement outcome with a certain object or an environment can be crucial to many organizations and businesses around the globe.

2.2 EXISTING SYSTEM

1. Surveys*:*

   User Engagement can be measured by conducting surveys where users will fill in a survey form to give information regarding their engagement levels.

2. Manual study from videos:

User Engagement is measured by a person by going through multiple videos and identifying the affective states of person.

2.2.1 LIMITATIONS OF THE EXISTING SYSTEM

1. Inaccurate
2. Less credible
3. Herculean task for a human to do all the analyzing
4. Practically impossible for vast amounts of data

2.3 PROPOSED SYSTEM

An automated engagement detection system that can analyze a person's engagement outcome with a certain object or an environment can be crucial to many organizations and businesses around the globe. Therefore, we employ cutting-edge algorithms in our project to recognize user engagement levels and divide them into two categories: positive and negative.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources, and the system will work in almost all configurations.

1. It is inexpensive in terms of effort and labor.
2. It is highly scalable.
3. Practically efficient and implementable

2.4 FEASIBILITY STUDY

A feasibility study is an analysis that considers all a project's relevant factors— including economic, technical, and social considerations—to ascertain the likelihood of completing the project successfully. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

Economic feasibility is a kind of cost-benefit analysis of the examined project, which assesses whether it is possible to implement it. This term means the assessment and analysis of a project's potential to support the decision-making process by objectively and rationally identifying its strengths, weaknesses, opportunities, and risks associated with it, the resources that will be needed to implement the project, and an assessment of its chances of success.

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

2.4.2 TECHNICAL FEASIBILITY

Technical feasibility is a set of techniques aimed at forecasting future prices of securities, currencies or raw materials based on the analysis of price formation in the past. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- System: 2vCPU @ 2.2GHz
- Hard Disk: 50 GB
- Input Devices: Keyboard, Mouse
- Ram: 8 GB

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system: Windows 8,10
- Coding Language: Python
- Tool: Google Colaboratory

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for anomaly detection using machine learning, starting from input to final prediction.

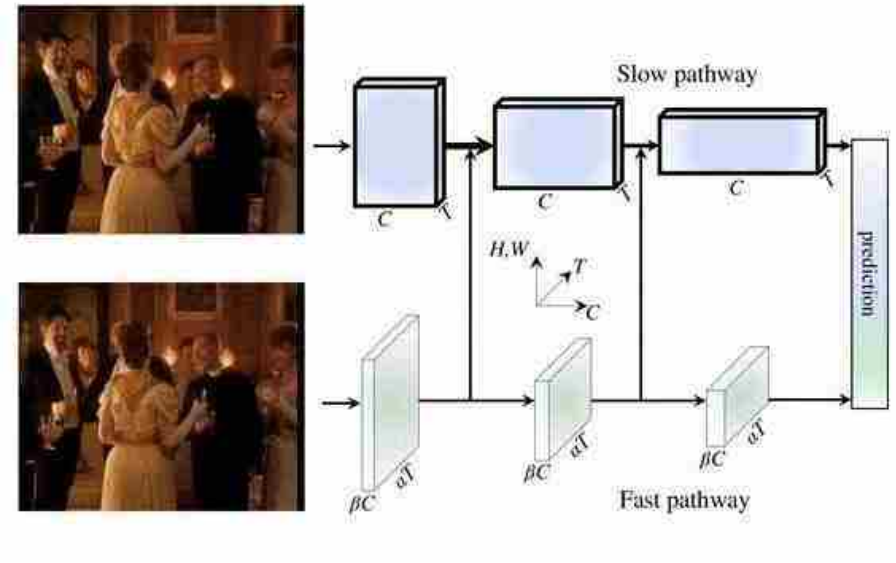


Figure 3.1: Project Architecture of Automated Engagement Recognition in E-environments

3.2 DESCRIPTION

Input Data: Input data is generally in video format where the data is read and described using graphs.

Reading Data: Pandas library is used to read the data from csv files.

Describing Data: In this following step we are going to describe the data in video file to know the number of rows and columns in the dataset.

Data Cleaning: It is a very important step while we are dealing with the large datasets. To achieve the efficiency in computation we are going to remove not related to crimevideos.

Training and test data: Training data is passed to train the model. Test data is used to test the trained model whether it is making correct predictions or not.

3.3 USE CASE DIAGRAM

In the use case diagram we have basically two actors who are the user and the admin. The user initiates the system to get the results. Whereas the admin login to the system , The Camera processes the images and the videos and installs it into the system which is then accessed by the admin and the system evaluates the user engagement and after all the analysis it gives the results.

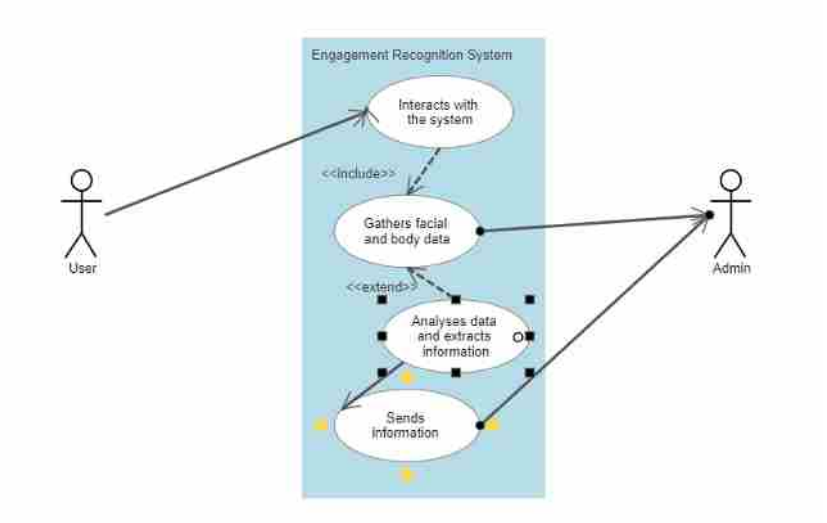


Figure 3.2: Use Case Diagram of Automated Engagement Recognition in E-environments

3.4 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

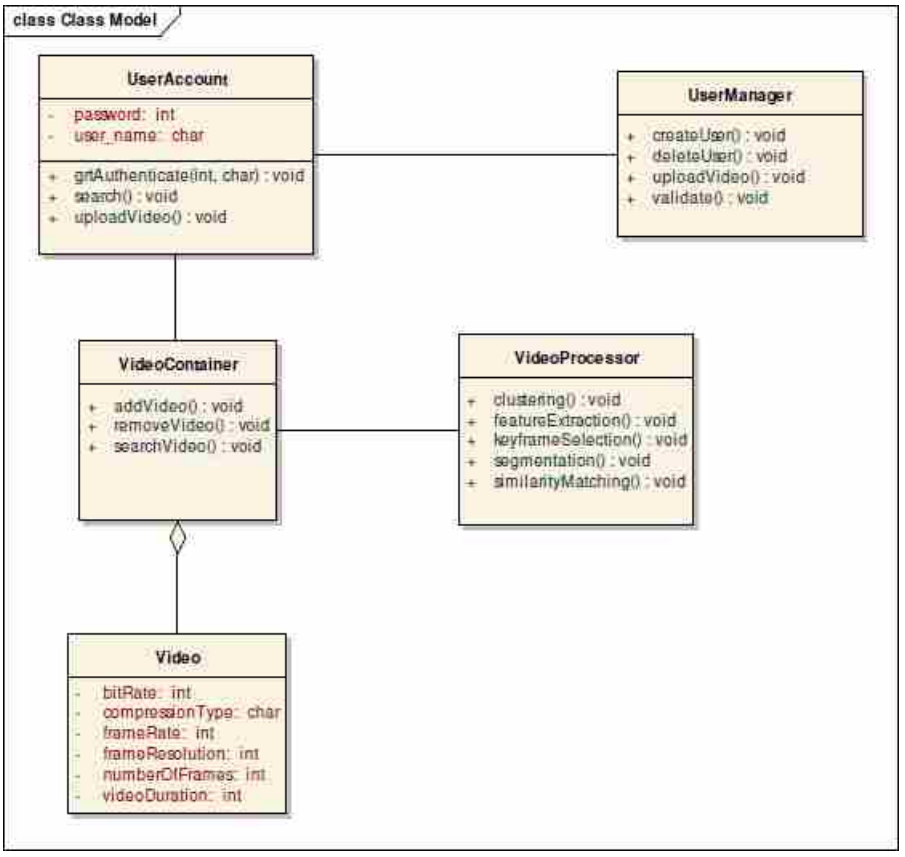
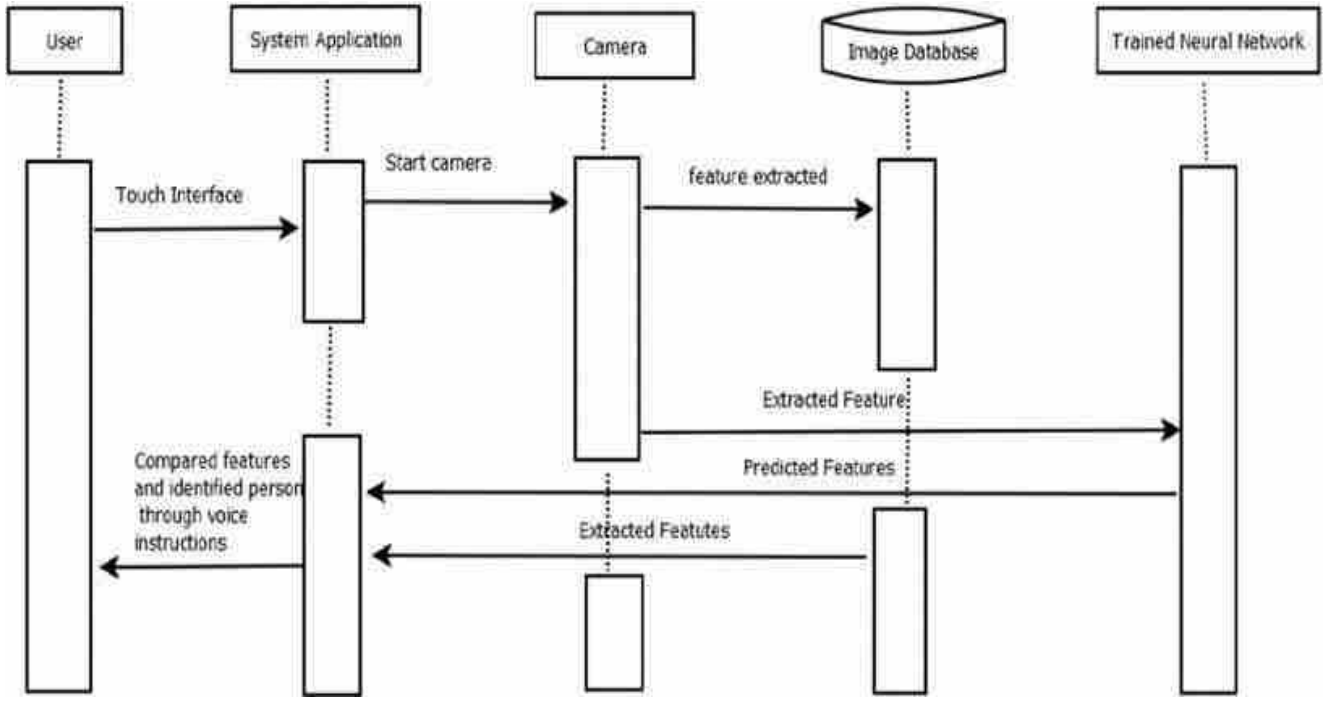


Figure 3.3: Class Diagram of Automated Engagement Recognition in E-environments

3.5 SEQUENCE DIAGRAM

Figure 3.4: Sequence Diagram of Automated Engagement Recognition in E-environments



3.6 ACTIVITY DIAGRAM

It describes the flow of activity states.

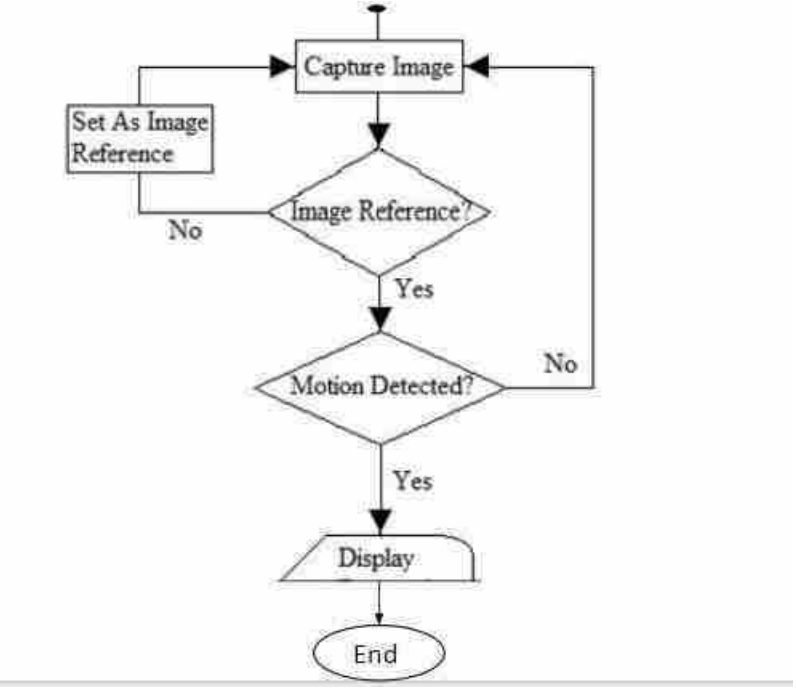


Figure 3.5: Activity Diagram of Automated Engagement Recognition in E-environments

4. IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

```
!nvidia-smi
from google.colab import drive drive._mount('/content/drive', force_remount= True)



# !sudo apt-get install -y python-dev pkg-config # !sudo apt-get install -y \
# libavformat-dev libavcodec-dev libavdevice-dev \
# libavutil-dev libswscale-dev libswresample-dev libavfilter-dev # !pip install av

# import av
# import glob # import os
# import time # import tqdm
# import datetime # import argparse

import os
os.listdir("/content/drive/Shareddrives/Manthan Data/Videos")



import cv2
import numpy as np import os
from google.colab.patches import cv2_imshow


def dataprep(path, seq_len, res):
for c in os.listdir(path):
if not os.path.exists(os.path.join(res,c)): os.makedirs(os.path.join(res,c))
for v in os.listdir(os.path.join(path, c)):



#
cap = cv2.VideoCapture(os.path.join(path, c, v))

# Get the frames per second
fps = cap.get(cv2.CAP_PROP_FPS)

# Get the total numer of frames in the video.
frame_count = cap.get(cv2.CAP_PROP_FRAME_COUNT)



skip = frame_count // seq_len frame_num = 0
```

```
count = 0
cap.set(cv2.CAP_PROP_POS_FRAMES, frame_num) # optional success, image = cap.read()
img1 = cv2.resize(image, (128,128)) while count < seq_len-1:
try:
frame_num += skip
cap.set(cv2.CAP_PROP_POS_FRAMES, frame_num) # optional success, image = cap.read()
image = cv2.resize(image, (128,128)) # print(frame_num)
except:
print(os.path.join(path, c,v))
img1 = np.append(img1,image,axis = 1) count += 1
cv2.imwrite(os.path.join(res,c,v)[:-4] +".jpg",img1)
print(os.path.join(res,c,v)[:-4] +".jpg") # frame_num = frame_number cv2_imshow(img1)
def flipdataprep(path, seq_len, res):
for c in os.listdir(path):
if not os.path.exists(os.path.join(res,c)): os.makedirs(os.path.join(res,c))
for v in os.listdir(os.path.join(path, c)):
frame_num = 0 #
cap = cv2.VideoCapture(os.path.join(path, c, v))

# Get the frames per second
fps = cap.get(cv2.CAP_PROP_FPS)

# Get the total numer of frames in the video.
frame_count = cap.get(cv2.CAP_PROP_FRAME_COUNT)

skip = frame_count // seq_len count = 0
cap.set(cv2.CAP_PROP_POS_FRAMES, frame_num) # optional success, image = cap.read()
img1 = cv2.resize(image, (128,128)) img1 = np.flip(img1,1)
while count < seq_len-1:
try:
frame_num += skip
cap.set(cv2.CAP_PROP_POS_FRAMES, frame_num) # optional success, image = cap.read()
image = cv2.resize(image, (128,128)) # print(frame_num)
except:
print(os.path.join(path, c,v)) # img1 = np.flip(img1,1) image = np.flip(image,1)
img1 = np.append(img1,image,axis = 1) count += 1
cv2.imwrite(os.path.join(res,c,v)[:-4] +"flip.jpg",img1) # frame_num = frame_number
cv2_imshow(img1)

#path to videos
path = "/content/drive/Shareddrives/Manthan Data/Videos" #path to destination
res = '/content/drive/Shareddrives/Manthan Data/frames/' #number of images per video
seq_len = 16
dataprep(path, seq_len , res)
# flipdataprep(path, seq_len, res) """### Training ##"""

import torchvision import torch
from torch import nn
import torch.nn.functional as F import torchvision.models as models
# from torchvision.models import Model import torch.optim as optim
import copy import os

from tqdm.autonotebook import tqdm import matplotlib.pyplot as plt
from torch.utils.data import Dataset from torchvision import transforms from torch.utils.data
import DataLoader import numpy as np
```

```
from torch.utils.data.sampler import SubsetRandomSampler import cv2
import sys
from torch.optim.lr_scheduler import StepLR
sys.path.append("/content/drive/Shareddrives/Manthan Data/") from model import *
# sys.path.append("/content/clr.py")

data_path = '/content/drive/MyDrive/IFrames/crime16' classes = os.listdir(data_path)
decoder = {}
for i in range(len(classes)):
decoder[classes[i]] = i encoder = {}
for i in range(len(classes)):
encoder[i] = classes[i] encoder

id = list()
path = '/content/drive/MyDrive/IFrames/crime16' for i in os.listdir(path):
p1 = os.path.join(path,i) for j in os.listdir(p1)[:500]:
p2 = os.path.join(p1,j) id.append((i,p2)) len(id)

class video_dataset(Dataset):
def init (self,frame_list,sequence_length = 16,transform = None): self.frame_list = frame_list
self.transform = transform self.sequence_length = sequence_length def len (self):
return len(self.frame_list) def getitem (self,idx):
label,path = self.frame_list[idx] img = cv2.imread(path) seq_img = list()
for i in range(16):
img1 = img[:,128*i:128*(i+1),:] if(self.transform):
img1 = self.transform(img1) seq_img.append(img1)
seq_image = torch.stack(seq_img)
seq_image = seq_image.reshape(3,16,im_size,im_size) return seq_image,decoder[label]

im_size = 128
mean = [0.4889, 0.4887, 0.4891]
std = [0.2074, 0.2074, 0.2074]


train_transforms = transforms.Compose([
transforms.ToPILImage(),transforms.RandomHorizontalFlip(),
transforms.RandomRotation(degrees=10),transforms.ToTensor(),
transforms.Resize((im_size,im_size)),])

train_data = video_dataset(id,sequence_length = 16,transform = train_transforms)



def mean_std_for_loader(loader: DataLoader):
# var[X] = E[X**2] - E[X]**2
channels_sum, channels_sqrd_sum, num_batches = 0, 0, 0 for data, _ in tqdm(loader):
this_batch_size = data.size()[0]
weight = this_batch_size / len(train_data)
channels_sum += weight*torch.mean(data, dim=[0, 2, 3])
channels_sqrd_sum += weight*torch.mean(data ** 2, dim=[0, 2, 3]) num_batches +=

weight
mean = channels_sum / num_batches
std = (channels_sqrd_sum / num_batches - mean ** 2) ** 0.5
mean = [np.mean(np.array(mean)[0]),np.mean(np.array(mean)[1]),np.mean(np.array(mean)[2])]
```

```
std = [np.mean(np.array(std)[0]),np.mean(np.array(std)[1]),np.mean(np.array(std)[2])] return
mean, std
# mean, std = mean_std_for_loader(train_data)

train_transforms = transforms.Compose([
transforms.ToPILImage(), transforms.RandomHorizontalFlip(),
transforms.RandomRotation(degrees=10), transforms.ToTensor(),
transforms.Resize((im_size,im_size)), transforms.Normalize(mean, std)])
train_data = video_dataset(id,sequence_length = 16,transform = train_transforms)
# train_loader = DataLoader(train_data,batch_size = 8,num_workers = 4 ,shuffle = True) #
dataloaders = {'train':train_loader}
print(mean, std)



# validation_split = 0.2 # shuffle_dataset = True # random_seed= 42
# batch_size = 8

# dataset_size = len(train_data)
# indices = list(range(dataset_size))
# split = int(np.floor(validation_split * dataset_size))
# if shuffle_dataset : #np.random.seed(random_seed) # np.random.shuffle(indices)
# train_indices, val_indices = indices[split:], indices[:split]

# # Creating PT data samplers and loaders:
# train_sampler = SubsetRandomSampler(train_indices) # valid_sampler =
SubsetRandomSampler(val_indices)
# train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
#sampler=train_sampler)
# validation_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
#sampler=valid_sampler)
# dataloaders = {"train": train_loader, "val": validation_loader} # mean = 0.0
# meansq = 0.0 # count = 0

# def mean_std_for_loader(loader: DataLoader):
#    # var[X] = E[X**2] - E[X]**2
#channels_sum, channels_sqrd_sum, num_batches = 0, 0, 0 # for data, _ in tqdm(loader):
#this_batch_size = data.size()[0] # weight = this_batch_size / 1
#channels_sum += weight*torch.mean(data, dim=[0, 2, 3])
# channels_sqrd_sum += weight*torch.mean(data ** 2, dim=[0, 2, 3]) # num_batches += weight
#mean = channels_sum / num_batches
#std = (channels_sqrd_sum / num_batches - mean ** 2) ** 0.5
#mean = [np.mean(np.array(mean)[0]),np.mean(np.array(mean)[1]),np.mean(np.array(mean)[2])]
#std = [np.mean(np.array(std)[0]),np.mean(np.array(std)[1]),np.mean(np.array(std)[2])] #return
mean, std
# mean_std_for_loader(train_data) # # len(train_data)

#
https://stackoverflow.com/questions/50544730/how-do-i-split-a-custom-dataset-into- training-
and-test-datasets/ 50544887#50544887
validation_split = 0.2 shuffle_dataset = True random_seed= 42
batch_size = 8

dataset_size = len(train_data) indices = list(range(dataset_size))
split = int(np.floor(validation_split * dataset_size)) if shuffle_dataset :
```

```
np.random.seed(random_seed) np.random.shuffle(indices)
train_indices, val_indices = indices[split:], indices[:split]

# Creating PT data samplers and loaders:
train_sampler = SubsetRandomSampler(train_indices) valid_sampler =
SubsetRandomSampler(val_indices)

train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
sampler=train_sampler)
validation_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
sampler=valid_sampler)
dataloaders
= {"train": train_loader, "val": validation_loader}
# from google.colab.patches import cv2_imshow # a = list(train_data)
# from matplotlib import pyplot as plt # for j in range(1,10):# # c = np.array(a[x])
# b = np.array(a[j][0]).reshape(16,3,im_size,im_size) # # print(b.shape)
# # print(b)
# # for i in range(16):
# for j in range(16):
# x = b[j,:,:,:]
#print(x.shape)
# x = np.reshape(x,(128,128,3)) #x = (x)
#plt.imshow(x) ## print(x.max()) #plt.show()

from model import resnet50
model = resnet50(class_num=10).to('cuda' if torch.cuda.is_available() else 'cpu')
# from clr import *
device = 'cuda' if torch.cuda.is_available() else 'cpu'

cls_criterion = nn.CrossEntropyLoss().to(device)
# optimizer = torch.optim.SGD(model.parameters(), lr=1e-3, momentum = 0.9,weight_decay =
1e-4) optimizer = torch.optim.Adam(model.parameters(), lr=0.01, betas=(0.9, 0.999), eps=1e-08,
weight_decay=0, amsgrad=False)
# optimizer = torch.optim.NAdam(model.parameters(), lr=0.002, betas=(0.9, 0.999), eps=1e-08,
weight_decay=0, momentum_decay=0.004)
num_epochs = 100
# onecyc = OneCycleLR(num_samples = len(train_loader),batch_size =
len(train_loader)*num_epochs,max_lr = 0.1, minimum_momentum = 1e-3)

# Commented out IPython magic to ensure Python compatibility. from sklearn.metrics import
accuracy_score os.makedirs('/content/weights_crime',exist_ok = True)
seed = 0 np.random.seed(seed) torch.manual_seed(seed) torch.cuda.manual_seed(seed)
from torch.autograd import Variable iteration = 0
acc_all = list() loss_all = list()
min_valid_loss = np.inf val_acc = []
v_acc = [] v_loss = []


scheduler = StepLR(optimizer, step_size=10, gamma=0.1, verbose= True) for epoch in

range(num_epochs):

print('')
print(f"--- Epoch {epoch} ---") phase1 = dataloaders.keys() for phase in phase1:
print('')
```

```
print(f"--- Phase {phase} ---")
epoch_metrics = {"loss": [], "acc": [],"val_loss": [], "val_acc": []} if phase == "train":


for batch_i, (X, y) in enumerate(dataloaders[phase]): #iteration = iteration+1
image_sequences = Variable(X.to(device), requires_grad=True)
labels = Variable(y.to(device), requires_grad=False) optimizer.zero_grad()
#model.lstm.reset_hidden_state()
predictions = model(image_sequences) loss = cls_criterion(predictions, labels)
# acc = 100 * (predictions.detach().argmax(1) == labels).cpu().numpy().mean()
_, preds = torch.max(predictions, 1) labels = labels.to("cpu")
preds = preds.to("cpu")
acc = 100 * accuracy_score(labels, preds)

# print((predictions.detach().argmax(1) == labels)) loss.backward()
optimizer.step() epoch_metrics["loss"].append(loss.item()) epoch_metrics["acc"].append(acc)
# if(phase=='train'):
# lr,mom =
# update_lr(optimizer, lr)
# update_mom(optimizer, mom)
batches_done = epoch * len(dataloaders[phase]) + batch_i batches_left = num_epochs *
len(dataloaders[phase]) - batches_done sys.stdout.write(
"\r[Epoch %d/%d] [Batch %d/%d] [Loss: %f (%f), Acc: %.2f%% (%.2f%%)]" #% (
epoch, num_epochs, batch_i,
len(dataloaders[phase]), loss.item(), np.mean(epoch_metrics["loss"]), acc,
np.mean(epoch_metrics["acc"]),
)
)

# Empty cache
 # if torch.cuda.is_available(): #torch.cuda.empty_cache() print("")
print('{} , acc: {}'.format(phase,np.mean(epoch_metrics["acc"])))
else:
with torch.no_grad():
valid_loss = 0.0
# Optional when not using Model Specific layer for data, labels in validation_loader:
# print(data.shape) # print(vlabels)
if torch.cuda.is_available():
data, labels = data.cuda(), labels.cuda() model.eval()
target = model(data)
loss = cls_criterion(target,labels) # print(loss)
valid_loss = loss.item() # print(valid_loss)
# val_acc = 100 * (target.detach().argmax(1) == labels).cpu().numpy().mean()
_, preds = torch.max(target, 1)
labels = labels.to("cpu") preds = preds.to("cpu")
 val_acc = 100 * accuracy_score(labels, preds) # print((target.detach().argmax()))

 epoch_metrics["val_loss"].append(valid_loss) epoch_metrics["val_acc"].append(val_acc) print('')
print('val_acc: {}'.format(np.mean(epoch_metrics["val_acc"]))) print('val_loss :

{}'.format(np.mean(epoch_metrics["val_loss"]))) if min_valid_loss >
np.mean(epoch_metrics["val_loss"]):
print(f'Validation Loss Decreased({min_valid_loss:.6f}---
>{np.mean(epoch_metrics["val_loss"]):.6f}) \t Saving The Model')
min_valid_loss = np.mean(epoch_metrics["val_loss"])
```

```
torch.save(model.state_dict(),'/content/weights_crime/c3d_{}_{}.h5'.format(epoch,str(np.mean(e
poch_metrics ["val_loss"]))[:4]))
if phase=='train':
acc_all.append(np.mean(epoch_metrics["acc"]))
loss_all.append(np.mean(epoch_metrics["loss"])) scheduler.step()
# optimser.step() if phase == "val":
v_acc.append(np.mean(epoch_metrics["val_acc"]))
v_loss.append(np.mean(epoch_metrics["val_loss"]))

"""##Inference"""

data_path = '/content/drive/MyDrive/IFrames/crime16' classes = os.listdir(data_path)
decoder = {}
for i in range(len(classes)):
decoder[classes[i]] = i encoder = {}
for i in range(len(classes)):
encoder[i] = classes[i]
id = list() test = []
path = '/content/drive/MyDrive/IFrames/crime16' # print(os.listdir(path))
for i in (os.listdir(path)):
p1 = os.path.join(path,i) # print(p1)
for j in (os.listdir(p1))[:1]:
p2 = os.path.join(p1,j) id.append((i,p2)) test.append(i)
id[:][:]

from model import resnet50
model = resnet50(class_num=8).to('cuda' if torch.cuda.is_available() else 'cpu')

# from clr import *
device = 'cuda' if torch.cuda.is_available() else 'cpu'

cls_criterion = nn.CrossEntropyLoss().to(device)
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3, momentum = 0.9,weight_decay = 1e-
4) num_epochs = 20
# onecyc = OneCycleLR(num_samples = len(train_loader),batch_size =
len(train_loader)*num_epochs,max_lr = 0.1, minimum_momentum = 1e-3)
```



```
import cv2
import numpy as np import os
from google.colab.patches import cv2_imshow # os.makedirs("normal/test/")
os.makedirs('normal/',exist_ok = True)

seq = 16
frame_num = 0

path = '/content/drive/MyDrive/Pro_data/Anomaly_Dataset/Anomaly_Videos/Anomaly-Videos-
Part-1/Abuse/ Abuse001_x264.mp4'
des = '/content/normal/'

cap = cv2.VideoCapture(path) # Get the frames per second
fps = cap.get(cv2.CAP_PROP_FPS)

# Get the total numer of frames in the video.
frame_count = cap.get(cv2.CAP_PROP_FRAME_COUNT) skip = 15
```

```
# print(frame_count)

# while frame_num + (seq * skip) <= frame_count: count = 0
frame_number = frame_num cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number) #
optional success, image = cap.read()
img1 = cv2.resize(image, (128,128)) while count < seq-1:
try:
frame_number += skip
cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number) # optional success, image = cap.read()
image = cv2.resize(image, (128,128)) except:
print(path)
# print(frame_num) # print(frame_count)
img1 = np.append(img1,image,axis = 1) count += 1
# frame_num += frame_number # k += 1

# cv2.imwrite(des + str(frame_number) +".jpg",img1) cv2.imwrite( "image.jpg",img1)
frame_num = frame_number cv2_imshow(img1)
img = img1

from IPython.display import HTML from base64 import b64encode

video_path = path
mp4 = open(video_path, "rb").read()data_url
= "data:video/mp4;base64," + b64encode(mp4).decode() HTML(f"""
<video width=400 controls>
<source src="{data_url}" type="video/mp4">
</video> """)

img = cv2.imread("/content/image.jpg") seq_img = list()
for i in range(16):
img1 = img[:,128*i:128*(i+1),:].astype("uint8")
img1 = torchvision.transforms.functional.to_tensor(img1) img1 =
transforms.Normalize(mean,std)(img1)
img1 = transforms.Resize((im_size,im_size))(img1)
img1 = img1.to('cuda' if torch.cuda.is_available() else 'cpu') seq_img.append(img1)
seq_image = torch.stack(seq_img)
seq_image = seq_image.reshape(3,16,im_size,im_size) seq_image = seq_image.reshape([1,3, 16,
128, 128])
# print(seq_image.shape)
# seq_image = seq_image.cuda()
from model import resnet50
model = resnet50(class_num=8).to('cuda' if torch.cuda.is_available() else 'cpu')

model.load_state_dict(torch.load('/content/drive/Shareddrives/One/c3d_11_0.01.h5',map_locatio
n=torch.device(device))) model.eval()
model(seq_image)
pred = (model(seq_image)).argmax() # tar.append(encoder[pred.item()])
print(encoder[pred.item()])

tar = []
for i in range(len(id)):
path = id[i][1] print(path)
if path.endswith(".jpg"):
```

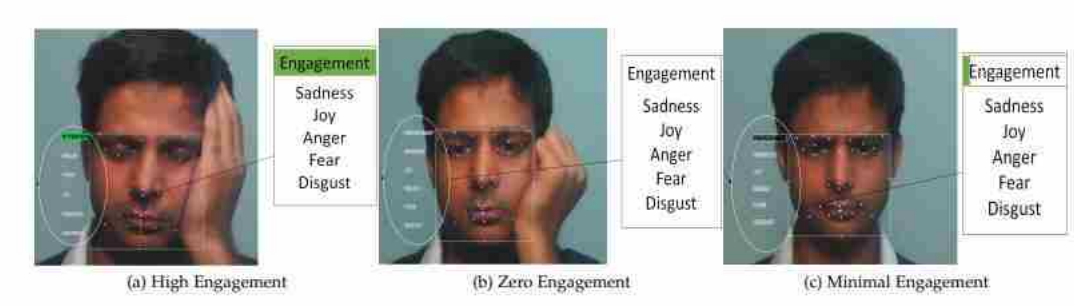
```
# print("ori",id[i][0]) img = cv2.imread(path) seq_img = list()
for i in range(16):
img1 = img[:,128*i:128*(i+1),:].astype("uint8")
img1 = torchvision.transforms.functional.to_tensor(img1) img1 =
transforms.Normalize(mean,std)(img1)
img1 = transforms.Resize((im_size,im_size))(img1)
img1 = img1.to('cuda' if torch.cuda.is_available() else 'cpu') seq_img.append(img1)
seq_image = torch.stack(seq_img)

seq_image = seq_image.reshape(3,16,im_size,im_size) seq_image = seq_image.reshape([1,3, 16,
128, 128])
# print(seq_image.shape)
# seq_image = seq_image.cuda()
from model import resnet50
model = resnet50(class_num=8).to('cuda' if torch.cuda.is_available() else 'cpu')
model.load_state_dict(torch.load('/content/drive/Shareddrives/One/c3d_11_0.01.h5',map_locatio
n=torch.device(device))) model.eval()
model(seq_image)
pred = (model(seq_image)).argmax() tar.append(encoder[pred.item()])
print(encoder[pred.item()])
```

5. SCREENSHOTS

5. SCREENSHOTS

5.1 EXPLORING THE DATASET



Screenshot 5.1: Engagement Analysis

Screenshot 5.2: Engagement level increasing from left to right

Screenshot 5.3: Variety in Dataset

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discovery conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows, data fields, predefined processes.

6.3 TEST CASES

6.3.1 UPLOADING IMAGES

Test case ID	Test case name	Purpose	Test Case	Output
1	User uploads videos	Use it for identification	The user uploads the positive engagement video for analysis	Uploaded successfully and positive analysis is generated
2	User uploads 2nd video	Use it for identification	The user uploads the negative engagement video for analysis	Uploaded successfully and negative analysis is generated

6.3.2 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1	Classification test 1	To check if the classifier performs its task	Positive engagement video	Positive analysis is generated.
2	Classification test 2	To check if the classifier performs its task	Negative engagement video	Negative analysis is generated.

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

Our work can accelerate the entire process of engagement detection using computer vision which will result in user analysis and business implementation. Evaluation results also indicate that the proposed implementation is effective in feature selection and prediction. This method can also be applied in other related research fields by fine tuning this existing method.

7.2 FUTURE SCOPE

1. Accuracy can be further improved to create a more robust model.
2. An application can be created to provide a comfortable UI/UX use case.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

1 SlowFast networks: https://arxiv.org/abs/1812.03982

2 Action Recognition: https://uwaterloo.ca/vision-image-processing-lab/research-demos/action-recognition-video

3 Human action recognition methods: https://www.frontiersin.org/articles/10.3389/frobt.2015.00028/full

4 Crime in India statistics: https://ncrb.gov.in/en/crime-india

5 Crime in India statistics: http://mospi.nic.in/sites/default/files/Statistical_year_book_india_chapters/ch37.pdf

6 Image and Video Understanding: https://medium.com/stradigiai/image-and-video-understanding-an-introduction-to-computer-vision-5d83f8fa63f5

7 Image/Video Understanding and Analysis: https://www.microsoft.com/en-us/research/project/image2text/

8 CNN for Deep Learning: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

A Comprehensive Guide to Convolutional Neural Networks: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53