Dayananda Sagar College of Engineering

Department of Electronics and Communication Engineering

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru – 560 078.

(An Autonomous Institute affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment and Accreditation Council (NAAC) with 'A' grade

Assignment

Program: B.E. Branch: ECE

Course: Machine Learning Semester : 7

Course Code: 19EC7DEMAL Date: 11/12/2023

A Report on

Drowsiness detection using LSTM + CNN

Submitted by

USN: 1DS20EC241 **NAME: Yash Anand**

1DS20EC191 **Siddhant Mohanta**

1DS18EC727 **Siddharth Shivam**

FACULTY IN-CHARGE

DR. K N PUSHPALATHA

Signature of Faculty In-charge

CONTENTS

1. Overview- Introduction on concept must be explained clearly
2. Flow chart/Algorithm
3. Program with Comments
4. Results and Discussion of all possible test cases(include few pics of execution)
5. References

1. OVERVIEW

The project aims to address the critical issue of driver drowsiness, a significant cause of road accidents worldwide. This code implements a hybrid model that combines Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) architectures for binary image classification using the Keras library, with a focus on computer vision tasks. The data generator function preprocesses training and validation images, enhancing their suitability for input to the model. The CNN component captures spatial features through convolutional, max-pooling, dropout, and fully connected layers, making it adept at image feature extraction. Simultaneously, the LSTM component processes sequential information, allowing the model to capture temporal dependencies within data. This integration of computer vision techniques and sequential learning enhances the model's capability for nuanced image classification. The model is compiled with the Adam optimizer and categorical crossentropy loss, and training is conducted using the `fit_generator` function. The resulting trained model can be saved for subsequent use, providing a flexible solution for a wide range of binary image classification tasks that involve both spatial and temporal considerations.

Link-https://github.com/Yash-1024/ML/tree/main/Drowziness%20Detection

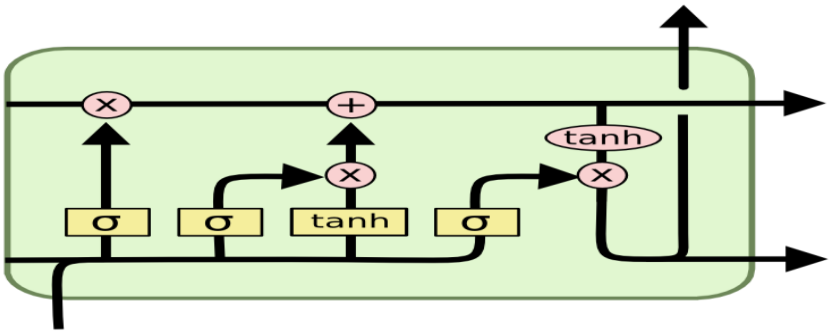
2. INTRODUCTION

LONG SHORT-TERM MEMORY (LSTM):

Full Form: LSTM stands for Long Short-Term Memory.

History: LSTM was introduced by Hochreiter & Schmidhuber in 1997 to address the vanishing gradient problem in traditional recurrent neural networks (RNNs). Its design was aimed at capturing long-term dependencies in sequential data, enabling better retention of information over time.

Network Architecture: LSTM networks consist of memory cells that regulate the flow of information. These cells contain gates that control the information flow, including input gates, forget gates, and output gates. Each gate is equipped to selectively remember or forget information, allowing LSTMs to retain crucial patterns over extended sequences.This architecture enables LSTMs to excel in tasks involving sequential data, such as natural language processing (NLP), time series prediction, and speech recognition.

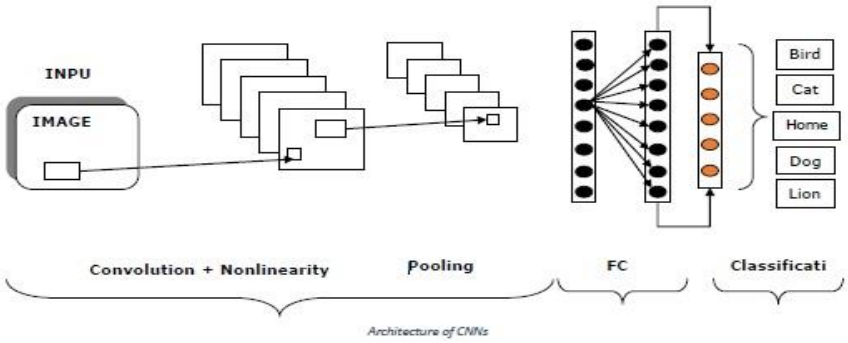


CONVOLUTIONAL NEURAL NETWORKS (CNN):

Full Form: CNN stands for Convolutional Neural Network.

History: CNNs were pioneered by Yann LeCun in the late 1980s for handwritten digit recognition tasks.The concept gained prominence after the development of LeNet-5 in 1998, primarily for recognizing handwritten digits in images.

Network Architecture: CNNs are structured with convolutional layers that employ filters to extract hierarchical features from input data.These networks use pooling layers to downsample feature maps, reducing computational load while retaining important spatial information. Fully connected layers at the end of the network perform classification/regression based on the learned features.CNNs are widely used in computer vision tasks, including image classification, object detection, and feature extraction, owing to their ability to effectively process visual data.

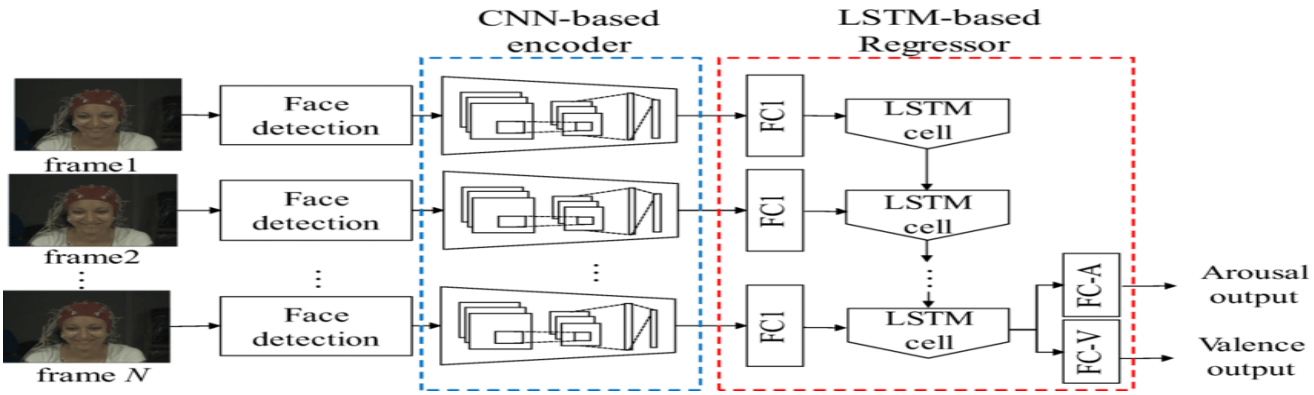


Architecture of CNNs

Evolution of Combination:

The integration of CNNs and LSTMs (CNN-LSTM or ConvLSTM) combines the strengths of both architectures. It enables the joint modeling of spatial and temporal dependencies in data, making it especially powerful in tasks that involve sequences within spatial structures, like video analysis, action

recognition, and spatiotemporal forecasting. This fusion has led to significant advancements in understanding and processing both sequential and spatial information.



3. FLOW CHART/ALGORITHM

ALGORITHM

- **Data Preparation:** Initially, images of alert and drowsy drivers are gathered and preprocessed. Facial recognition methods like Haar Cascades are applied to identify and extract facial regions, ensuring standardized input by resizing and cropping these regions.

- **CNN Feature Extraction:** The system's core architecture is a CNN, leveraging convolutional layers for hierarchical feature extraction from the facial images. These layers identify essential patterns related to drowsiness, such as eye closure or facial expressions. Dropout layers help avoid overfitting, and dense layers perform the classification task, determining the driver's state (alert or drowsy) based on the learned features.

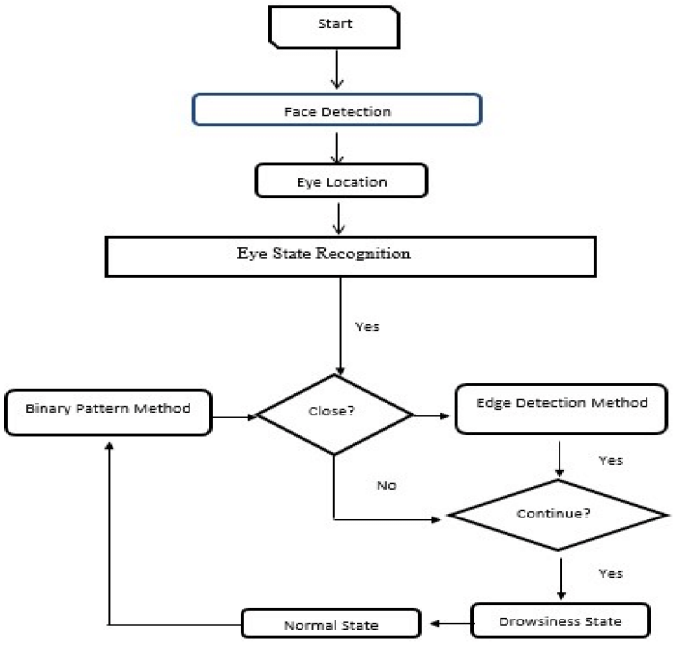
- **LSTM Integration:** Although the code provided doesn't explicitly incorporate LSTMs, integrating them could significantly improve performance. LSTMs are adept at capturing temporal dependencies across consecutive facial frames, offering a nuanced understanding of drowsiness indicators evolving over time. This integration could bolster the system's accuracy in detecting drowsy drivers.

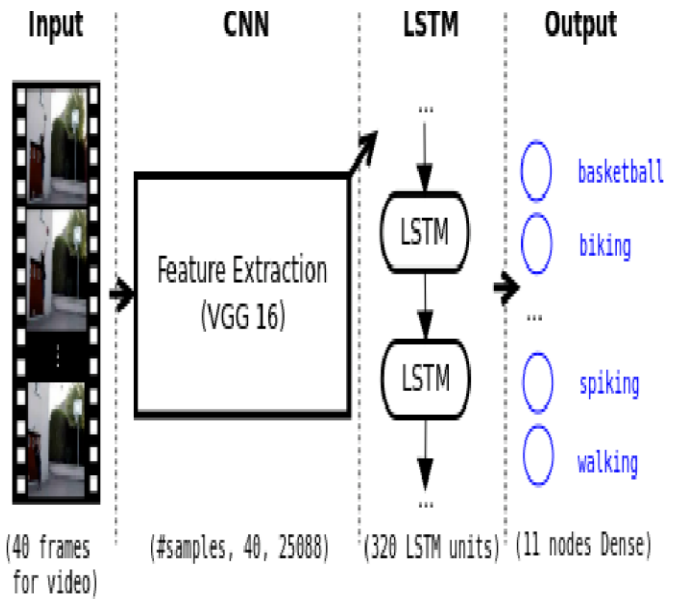
- **Data Augmentation:** To enhance model robustness, data augmentation techniques are utilized, varying images to augment the training dataset. The model is trained using an Adam optimizer and categorical cross-entropy loss to classify images accurately. Training progress is tracked through visualizations displaying accuracy and loss over epochs.

- **Model Architecture:** The architecture builds upon the CNN architecture and is typically comprised of thick layers, pooling layers, and convolutional layers. It utilizes ReLU and other activation functions within hidden layers, while Softmax may be employed in the output layer for closed or open multi-class categorization tasks.

- **Training:** This model is trained on a dataset of pictures with open and closed eyes. It uses loss functions and optimization strategies, such as the Adam optimizer and categorical cross-entropy, to minimize the error between its predictions and the actual labels. The model cycles through epochs, refining its predictions with each pass and progressively improving its accuracy.
- **Prediction:** Trained on a dataset of eye pictures, this model acts like a fortune teller for your eyelids. It analyses preprocessed images of your eyes and, based on the patterns it has learned, predicts whether they will be open or closed next. Instead of a crystal ball, it uses sophisticated algorithms to churn out probabilities for each state, giving you a sneak peek into the future of your blinks.
- **Threshold-based Decision Making:** The system assesses eye state probabilities to classify drowsiness levels, activating alerts based on thresholds for closed eyes, slight drowsiness, wakefulness, and alertness.

BLOCK DIAGRAM





4. PROGRAM

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time
mixer.init()
sound = mixer.Sound('alarm.wav')
face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')
lbl=['Close','Open']
model = load_model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred=[99]
lpred=[99]
while(True):
```

```
ret, frame = cap.read()
height,width = frame.shape[:2]
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
left_eye = leye.detectMultiScale(gray)
right_eye =  reye.detectMultiScale(gray)
cv2.rectangle(frame, (0,height-50) , (200,height) , (0,0,0) , thickness=cv2.FILLED )
for (x,y,w,h) in faces:
    cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )
for (x,y,w,h) in right_eye:
    r_eye=frame[y:y+h,x:x+w]
    count=count+1
    r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
    r_eye = cv2.resize(r_eye,(24,24))
    r_eye= r_eye/255
    r_eye=  r_eye.reshape(24,24,-1)
    r_eye = np.expand_dims(r_eye,axis=0)
    rpred = model.predict(r_eye)
    print(rpred)
    if(np.argmax(rpred)==1):
        lbl='Open'
    if(np.argmax(rpred)==0):
        lbl='Closed'
    break
```

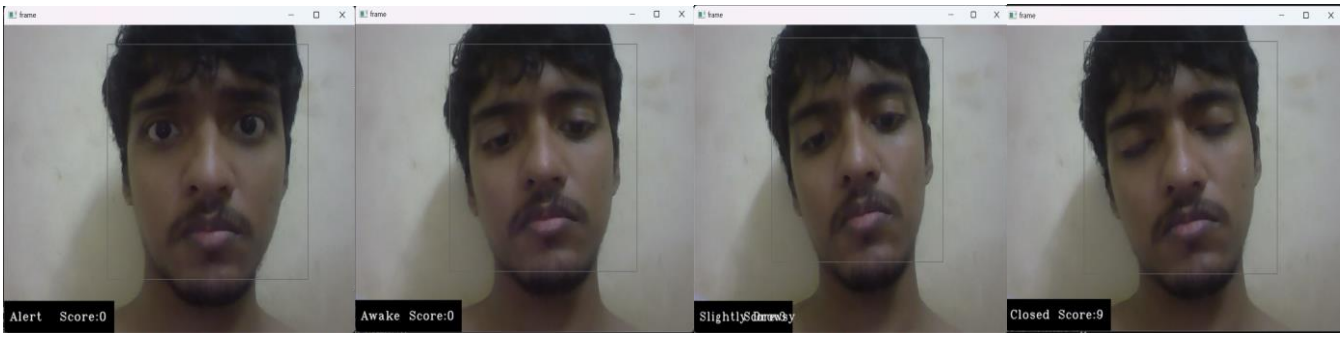
```
for (x,y,w,h) in left_eye:
    l_eye=frame[y:y+h,x:x+w]
    count=count+1
    l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
    l_eye = cv2.resize(l_eye,(24,24))
    l_eye= l_eye/255
    l_eye=l_eye.reshape(24,24,-1)
    l_eye = np.expand_dims(l_eye,axis=0)
    lpred = model.predict(l_eye)
    print(lpred)
    if(np.argmax(lpred)==1):
        lbl='Open'
    if(np.argmax(lpred)==0):
        lbl='Closed'
    break
if(np.argmax(rpred)==0 and np.argmax(lpred)==0):
    score=score+1
    cv2.putText(frame,"Closed",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
# if(rpred[0]==1 or lpred[0]==1):
else:
    score=score-1
    cv2.putText(frame,"Open",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
if(score<0):
    score=0
```

```
    cv2.putText(frame,'Score:'+str(score),(100,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
    if(score>15):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            sound.play()
        except:  # isplaying = False
            pass
        if(thicc<16):
            thicc= thicc+2
        else:
            thicc=thicc-2
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

5. RESULT

Employing real-time eye state analysis, the system adeptly distinguishes between closed, slightly drowsy, awake, and alert conditions. With predefined thresholds for each state, it triggers timely alerts, mitigating potential accidents linked to drowsiness. This proactive mechanism significantly enhances safety, especially in critical contexts like driving or monitoring, where sustained vigilance is paramount. By promptly addressing lapses in attention and fatigue-related risks, the system ensures swift intervention,

fostering an environment conducive to preventing hazards arising from reduced alertness. Ultimately, it contributes substantially to averting potential accidents due to compromised levels of wakefulness.



Strengths:

- Effectively discerns between various eye states, enabling precise detection of drowsiness levels.
- Offers timely alerts, crucial in critical scenarios like driving, enhancing safety.
- Prevents potential accidents by addressing lapses in vigilance and alertness promptly.

Weaknesses:

- Might not consider other fatigue indicators beyond eye states, potentially missing overall drowsiness cues.
- Accuracy tied to proper eye detection, susceptible to variations in lighting or occlusions.
- Predefined thresholds might not universally apply to all individuals, impacting accuracy.

Future Work:

- Enhance the system to accommodate diverse environmental conditions, accommodating variations in lighting, camera angles, and user diversity.
- Integrate additional sensor data, such as head pose estimation or behavioral cues, to improve accuracy and expand the scope of drowsiness detection.
- Implement automated interventions, like adaptive alarms or prompts, to actively engage and alert users during detected drowsiness, enhancing safety measures.

Conclusion:

In summary, this project showcases a valuable system for real-time drowsiness detection, especially in safety-critical scenarios like driving. Its accurate classification and prompt alerts offer proactive safety measures. However, limitations in contextual awareness and dependency on precise eye detection exist. Nevertheless, its swift intervention to mitigate drowsiness-related risks presents a significant advancement in safety technology. Future improvements addressing contextual variations could enhance its effectiveness for broader safety applications.

6.REFERENCE

- "Driver Drowsiness Detection Using Facial Parameters and RNNs with LSTM": https://www.researchgate.net/publication/339765089_Real-Time_System_for_Driver_Fatigue_Detection_Based_on_a_Recurrent_Neuronal_Network
- "Asleep at the Wheel: A Computer Vision and Deep Learning Approach to Detecting Drowsiness":
- https://medium.com/@jaynishvaghela/driver-drowsiness-detection-using-lstm-network-2966c49a5400
- "Algorithms-for-Drowsy-Driver-Detection": https://github.com/nishagandhi/DrowsyDriverDetection
- "LSTM-CNN model of drowsiness detection from multiple consciousness states acquired by EEG": https://ieeexplore.ieee.org/document/9669234
- 5.National Sleep Foundation: https://www.sleepfoundation.org/
- 6.National Highway Traffic Safety Administration: https://www.nhtsa.gov/
- 6.Drowsiness Detection Research Projects: https://www.researchgate.net/publication/327539984_Using_Image_Processing_in_the_Proposed_Drowsiness_Detection_System_Design