بسمه تعالی

تشخیص خواب آلودگی راننده با تمام المان ها

دانشجو:

علی صادقی فر

استاد:

دکتر ذبیحی فر

هوش مصنوعی

دانشکده مهندسی مکانیک

زمستان 1403

مقدمه

خواب‌آلودگی یکی از مهم‌ترین عوامل تصادفات جاده‌ای در سراسر جهان است. بر اساس آمارهای جهانی، درصد قابل‌توجهی از تصادفات منجر به مرگ یا جراحات شدید ناشی از خستگی و خواب‌آلودگی رانندگان است. این مشکل به ویژه در رانندگی‌های طولانی‌مدت و در شب‌ها شدت بیشتری پیدا می‌کند. تشخیص به موقع خواب‌آلودگی راننده می‌تواند تأثیر بسزایی در کاهش تصادفات و حفظ جان انسان‌ها داشته باشد.

اهمیت تشخیص خواب‌آلودگی راننده

خواب‌آلودگی راننده می‌تواند منجر به کاهش هوشیاری، واکنش‌های کندتر، و حتی از دست دادن کنترل خودرو شود. به همین دلیل، توسعه سیستم‌های هوشمندی که قادر به تشخیص این حالت باشند، از اهمیت بالایی برخوردار است. چنین سیستم‌هایی می‌توانند به راننده هشدار دهند یا حتی خودرو را به طور خودکار متوقف کنند تا از وقوع حوادث جلوگیری شود.

مروری بر روش‌های موجود

روش‌های مختلفی برای تشخیص خواب‌آلودگی پیشنهاد شده‌اند که می‌توان آنها را به سه دسته اصلی تقسیم کرد:

1. روش‌های مبتنی بر رفتار راننده

این روش‌ها حرکات سر، چشم، و حالات چهره راننده را تحلیل می‌کنند. به عنوان مثال، طولانی شدن زمان بسته بودن چشم‌ها می‌تواند نشانه خواب‌آلودگی باشد.

2. روش‌های مبتنی بر فیزیولوژی

EEG(الکتروانسفالوگرافی)، ECG(الکتروکاردیوگرافی) و EMG(الکترومایوگرافی)

3.روش‌های مبتنی بر عملکرد خودرو

تغییرات در الگوی حرکت خودرو، مانند انحراف از مسیر، تغییرات غیرعادی سرعت، و ترمزهای ناگهانی در این دسته قرار می‌گیرند.

هرچند روش‌های مبتنی بر فیزیولوژی دقیق‌تر هستند، اما به تجهیزات پیچیده نیاز دارند و برای استفاده در خودروهای شخصی مناسب نیستند. در مقابل، روش‌های مبتنی بر رفتار راننده به دلیل استفاده از دوربین‌های ارزان‌قیمت و الگوریتم‌های پردازش تصویر، گزینه‌ای کاربردی و عملی‌تر به شمار می‌آیند.

هدف پروژه

هدف این پروژه توسعه یک سیستم هوشمند تشخیص خواب‌آلودگی راننده با استفاده از شبکه‌های عصبی است. در این پروژه از تصاویر چهره افراد مختلف استفاده شده و با تحلیل ویژگی‌های صورت آنها میتوان به مدلی با دقت مطلوب دست یافت.

دیتاست مورد استفاده

مجموعه داده های مورد استفاده در پروژه از **kaggle** گرفته شده است. بیش از 41000 عکس با دو کلاس خواب آلود و غیر خواب آلود می باشد.

لینک صفحه:

https://www.kaggle.com/datasets/ismailnasri20/driver-drowsiness-dataset-ddd

کد آموزش مدل شبکه عصبی

```
import os
import cv2
import numpy as np
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

# 1. تنظیمات و پارامترها

from google.colab import drive
drive.mount('/content/drive')

IMAGE_SIZE = (64, 64)  # اندازه تصاویر ورودی به مدل
DATASET_PATH = "/content/drive/My Drive/Drowsiness Status"  # مسیر دیتاست تصاویر
CATEGORIES = ["Not_Sleepy", "Sleepy"]  # دسته‌بندی‌ها
```

ابتدا ماژول های مورد استفاده را ایمپورت میکنیم و مسیر دیتاست که در گوگل درایو ذخیره شده را وارد میکنیم.

```
# بارگذاری و پردازش تصاویر
def load_dataset():
    images = []
    labels = []
    features = []

    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor("/content/drive/MyDrive/shape_predictor_68_face_landmarks.dat")

    for label, category in enumerate(CATEGORIES):
        category_path = os.path.join(DATASET_PATH, category)
        for img_name in os.listdir(category_path):
            img_path = os.path.join(category_path, img_name)
            img = cv2.imread(img_path)
            if img is not None:
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                faces = detector(gray)

                for face in faces:
                    landmarks = predictor(gray, face)
                    left_eye = (landmarks.part(36).x, landmarks.part(36).y)
                    right_eye = (landmarks.part(45).x, landmarks.part(45).y)
                    mouth = (landmarks.part(62).x, landmarks.part(62).y)
```

تعریف Features برای ذخیره ویژگی‌های استخراج‌شده از چهره (مانند فاصله بین چشم‌ها و دهان)

🔷 detector: مدل تشخیص چهره

🔷 predictor: پیشبینی نقاط کلیدی چهره

فایل shape_predictor_68_face_landmarks.dat:

یک مدل آموزش دیده است که برای تشخیص نقاط روی چهره استفاده می‌شود. این مدل توسط **Dlib** ارائه شده است.

برای این موارد میتوان از این مدل استفاده کرد:

- ✅ **تشخیص احساسات**
- ✅ **ردیابی چشم و تشخیص خستگی یا خواب‌آلودگی**
- ✅ **پردازش چهره و زیبایی‌شناسی (مثل فیلترهای اینستاگرام)**
- ✅ **انیمیشن‌سازی و واقعیت افزوده (AR)**

🔷 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

تبدیل تصویر به سیاه و سفید

faces = detector(gray)

چهره‌ها را در تصویر جستجو کرده و مختصات آن‌ها را برمی‌گرداند

```
    for label, category in enumerate(CATEGORIES):
        category_path = os.path.join(DATASET_PATH, category)
        for img_name in os.listdir(category_path):
            img_path = os.path.join(category_path, img_name)
            img = cv2.imread(img_path)
            if img is not None:
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                faces = detector(gray)

                for face in faces:
                    landmarks = predictor(gray, face)
                    left_eye = (landmarks.part(36).x, landmarks.part(36).y)
                    right_eye = (landmarks.part(45).x, landmarks.part(45).y)
                    mouth = (landmarks.part(62).x, landmarks.part(62).y)

                    img = cv2.resize(img, (IMAGE_SIZE, IMAGE_SIZE))
                    img = img / 255.0  # نرمال‌سازی

                    images.append(img)
                    labels.append(label)
                    features.append([left_eye[0] - right_eye[0], mouth[1] - left_eye[1]])
                    break  # یک چهره در تصویر کافی است

    return np.array(images), np.array(features), np.array(labels)
```

نقاط کلیدی چهره مثل دو چشم و دهان را بررسی و مختصات آن ها را استخراج میکند.

```
# بارگذاری داده‌ها
X_images, X_features, y = load_dataset()
y = to_categorical(y, num_classes=len(CATEGORIES))

# تقسیم داده‌ها
X_train_images, X_test_images, X_train_features, X_test_features, y_train, y_test = train_test_split(
    X_images, X_features, y, test_size=0.2, random_state=42)

# مدل CNN برای تصاویر
image_input = Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3))
x = Conv2D(32, (3, 3), activation='relu')(image_input)
x = MaxPooling2D(2, 2)(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D(2, 2)(x)
x = Flatten()(x)

# مدل MLP برای ویژگی‌ها
feature_input = Input(shape=(2,))  # دو ویژگی عددی
y = Dense(32, activation='relu')(feature_input)
```

داده ها به دو دسته آموزش و تست تقسیم میشوند و 80 درصد برای آموزش و 20 درصد برای تست استفاده می شوند.

:random_state=42

مقدار ثابت برای قابل تکرار بودن آزمایش‌ها

مدل های CNN و MLP را تعریف میکنیم.

```
# ترکیب دو مدل
combined = concatenate([x, y])
output = Dense(len(CATEGORIES), activation='softmax')(combined)

model = Model(inputs=[image_input, feature_input], outputs=output)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# آموزش مدل
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
model.fit([X_train_images, X_train_features], y_train, epochs=10, batch_size=32, validation_data=([X_test_images, X_test_features], y_test), callbacks=[early_stopping])
```

دو مدل را با هم ترکیب میکنیم و از این موارد استفاده میکنیم:

ورودی مدل: شامل تصاویر و ویژگی‌های عددی

بهینه ساز برای یادگیری بهتر مدل (adam)

تابع هزینه برای طبقه‌بندی چندکلاسه (categorical_crossentropy)

تعریف **Early Stopping** برای جلوگیری از **overfitting**

اگر خطای داده‌های اعتبارسنجی برای 3 دوره متوالی بهبود نیابد، آموزش را متوقف کند.

```
# 7. ارزیابی مدل
loss, accuracy = model.evaluate(X_test_images, y_test)
print(f"دقت مدل: {accuracy * 100:.2f}%")

# 8. ذخیره مدل
model.save("/content/drive/My Drive/sleep_detection_model.h5")
print("مدل ذخیره شد.")
```

عملکرد مدل روی داده های تست ارزیابی می شود و میزان خطا و دقت محاسبه می شود.

استفاده از مدل آموزش دیده

```
import cv2
import numpy as np
import tensorflow as tf

# تنظیمات مدل و اندازه تصویر
IMAGE_SIZE = (64, 64)
CATEGORIES = ["Not_Sleepy", "Sleepy"]

# بارگذاری مدل
model = tf.keras.models.load_model("/content/drive/My Drive/sleep_detection_model.h5")
print("Model loaded successfully!")

# تابع پیش‌بینی
def predict_frame(frame):
    if frame is None or frame.size == 0:
        raise ValueError("Error: The input frame is empty or None!")
    try:
        img = cv2.resize(frame, IMAGE_SIZE)
        img = img / 255.0
        img = np.expand_dims(img, axis=0)
        prediction = model.predict(img)
        class_index = np.argmax(prediction)
        confidence = prediction[0][class_index] * 100
        return CATEGORIES[class_index], confidence
    except Exception as e:
        print(f"Error during prediction: {e}")
        return None, None
```

مدل ذخیره شده را بارگذاری میکنیم و تابع پیش بینی را تعریف میکنیم.

تصویر ورودی را به **ابعاد** ۶۴×۶۴ تغییر می‌دهد تا با مدل سازگار باشد.
مقدار پیکسل‌ها را **بین** ۰ تا ۱ نرمال‌سازی می‌کند.

بعد اضافی به آرایه اضافه می‌شود تا شکل آن برای ورودی مدل مناسب شود.

مدل روی تصویر پردازش کرده **و** کلاسی را که بیشترین احتمال را دارد، انتخاب می‌کند.
میزان اطمینان مدل نیز درصدی از خروجی محاسبه شده و همراه با کلاس برگردانده می‌شود.

```
# اتصال به دوربین
cap = cv2.VideoCapture(0)
if cap.isOpened():
    print("Video stream opened successfully!")
else:
    print("Error: Unable to open video stream")

# پردازش و نمایش تصویر
while cap.isOpened():
    ret, frame = cap.read()
    if not ret or frame is None:
        print("Error: Failed to capture frame")
        break

    # پیش‌بینی
    result, confidence = predict_frame(frame)
    if result is not None:
        cv2.putText(frame, f"{result} ({confidence:.2f}%)", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    # نمایش تصویر
    cv2.imshow("Live Camera", frame)

    # خروج با زدن کلید 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# آزادسازی دوربین
cap.release()
cv2.destroyAllWindows()
```

اتصال به دوربین:

به دوربین پیش فرض(در اینجا وبکم لپتاپ) متصل می شود. با استفاده از نرم افزار هایی مانند IP Webcam میتوان به دوربین گوشی هم متصل شد.

پردازش و نمایش تصویر:

حلقه تعریف شده تصاویر را از دوربین به صورت فریم به فریم دریافت می‌کند

هر فریم به تابع predict_frame ارسال می شود تا نتیجه و میزان اطمینان پیش بینی محاسبه شود.

خروج با فشردن "q":

با فشردن q برنامه خاتمه میابد.

آزادسازی منابع:

در پایان، دوربین و منابع مرتبط آزاد شده و تمام پنجره‌های باز بسته می‌شوند.

رسم نمودار دقت و خطا(در صورت نیاز)

```
import matplotlib.pyplot as plt

# رسم نمودار دقت
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# رسم نمودار خطا
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```