Wykrywanie zmęczenia u pracowników biurowych z wykorzystaniem uczenia maszynowego.

1. Wstęp.

Według różnych szacowań nawet do 20% wszystkich wypadków spowodowane jest zmęczeniem kierowcy.

Nic zatem dziwnego, że odkąd w 2007 szwedzka firma samochodowa Volvo po raz pierwszy wprowadziła na rynek system wykrywania zmęczenia u kierowców, ich popularność błyskawicznie zaczęła rosnąć.

Jest oczywiste, że technologia, która dosłownie ratuje nasze zdrowie oraz życie zasługuje na uwagę i dzisiaj znaczna część społeczeństwa przynajmniej słyszała o wykrywaniu senności u kierowców.

W tej pracy chciałbym jednak zwrócić uwagę na inny problem.

W dzisiejszych czasach coraz więcej ludzi boryka się z brakiem balansu między pracą a życiem prywatnym, często ciężko jest nam samodzielnie wyznaczać sobie rozsądne granice i w rezultacie przepracowujemy się.

Taka sytuacja wpływa negatywnie na nasze zdrowie psychiczne, a ponadto w dłuższej perspektywie okazuje się, że wcale nie zwiększamy naszej produktywności, gdyż pod wpływem zmęczenia nie wykonujemy swoich obowiązków w optymalny sposób.

Poniżej przedstawię program wykorzystujący sztuczną inteligencję, który może posłużyć do wykrywania i analizy zmęczenia za pomocą kamerki internetowej.

2. Założenia

Czynności, które postanowiłem wziąć pod uwagę przy ocenianiu zmęczenia to:

- ziewanie

- zamykanie oczu dłużej niż określona liczba sekund

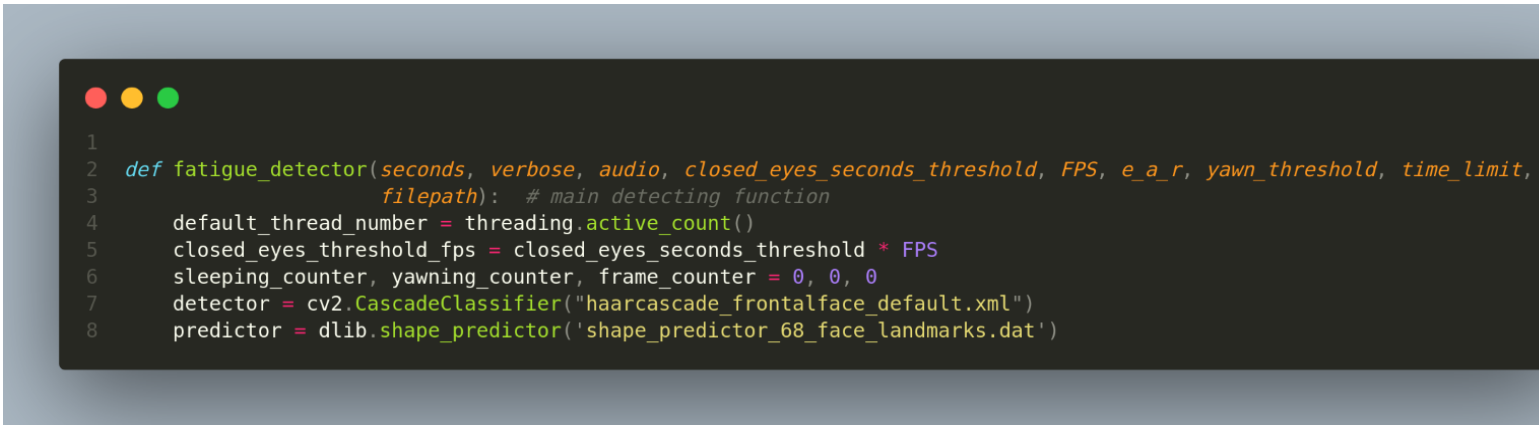
3. Omówienie programu

Kod dostępny jest pod tym linkiem: [Fatigue Recognition Project]

3.1

Program korzysta z dwóch niezastąpionych narzędzi, bez których zrealizowanie projektu

byłoby o wiele trudniejsze i bardziej czasochłonne, zatem zamierzam je pokrótce omówić.



```
def fatigue_detector(seconds, verbose, audio, closed_eyes_seconds_threshold, FPS, e_a_r, yawn_threshold, time_limit,
                     filepath):  # main detecting function
    default_thread_number = threading.active_count()
    closed_eyes_threshold_fps = closed_eyes_seconds_threshold * FPS
    sleeping_counter, yawning_counter, frame_counter = 0, 0, 0
    detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```

Jak widać główna funkcja zawiera w sobie między innymi takie zmienne:

- detector (detektor)

- predictor (predykcja)

Obie korzystają ze specjalnych plików i funkcji:

- cv2.CascadeClassifier:

- funkcja wykorzystująca metodę opartą na uczeniu maszynowym, w której funkcja kaskadowa jest trenowana na podstawie wielu obrazów. Następnie jest ona wykorzystywana do wykrywania obiektów na innych obrazach.
- plik "haarcascade_frontalface_default.xml" zawiera model, na podstawie którego powyższa funkcja wykrywa twarz

- dlib.shape_predictor:

- funkcja umożliwiającą wykrycie i podanie koordynatów danego kształtu
- plik " shape_predictor_68_face_landmarks.dat " zawiera model, umożliwiający określenie położenie czy też pozycje danego obiektu lub jego elementów (w tym przypadku twarzy)

Dzięki nim można w czasie rzeczywistym wykryć za pomocą kamerki twarz, a także poszczególne elementy fizjonomii jak np. usta i oczy.

3.2

Kiedy jesteśmy już w stanie zlokalizować usta oraz oczy, czas na następny krok, a mianowicie wyliczenie stosunku ich wysokości do szerokości, co umożliwi nam sprawdzanie, czy w danym momencie osoba ziewa lub zamyka oczy.

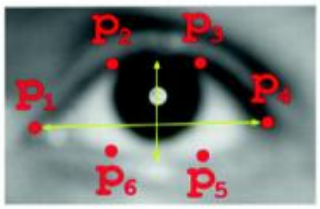
Odpowiedzialny jest za to poniższy kod:

```python
def eye_aspect_ratio(eye):  # calculates openness of an eye
    eye_height_line_1 = dist.euclidean(eye[1], eye[5])
    eye_height_line_2 = dist.euclidean(eye[2], eye[4])
    eye_width = dist.euclidean(eye[0], eye[3])
    ear = (eye_height_line_1 + eye_height_line_2) / (2.0 * eye_width)
    return ear


def final_ear(shape):
    (left_eye_start, left_eye_end) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]  # d
    (right_eye_start, right_eye_end) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    left_eye = shape[left_eye_start:left_eye_end]
    right_eye = shape[right_eye_start:right_eye_end]
    left_ear = eye_aspect_ratio(left_eye)
    right_ear = eye_aspect_ratio(right_eye)
    ear = (left_ear + right_ear) / 2.0
    return ear, left_eye, right_eye


def lip_distance(shape):  # calculates openness of a mouth
    top_lip = np.concatenate((shape[50:53], shape[61:64]))
    low_lip = np.concatenate((shape[56:59], shape[65:68]))
    top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)
    distance = abs(top_mean[1] - low_mean[1])
    return distance
```

Sposób działania pierwszych dwóch funkcji można zobrazować za pomocą poniższej ilustracji:



$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Jak widać, żeby obliczyć stosunek otwarcia oczu bierzemy średnią z dwóch pionowych linii stycznych do źrenicy i dzielimy przez linię idącą od jednego kącika oka do drugiego.

Z kolei do wyliczenia otwarcia ust wyciągamy średnią z długości dolnej i górnej wargi, dzięki czemu uzyskujemy parę punktów mniej więcej na środku obu warg, wtedy wystarczy odjąć je od siebie i wziąć wartość bezwzględną z tej wartości.

Te operacje wykonujemy w pętli i aktualizujemy nasze dane, co każdą klatkę, dzięki czemu program skutecznie i niemal błyskawicznie wykrywa objawy zmęczenia.

4. Dodatkowe funkcjonalności

Program można skonfigurować na wiele różnych sposobów, tak żeby dopasować go do swoich potrzeb. W tym celu nie trzeba ingerować w kod, wystarczy podać poszczególne flagi/argumenty w wierszu poleceń.

Są to:

-r, (rounds), ile razy program ma się wykonać

-s (seconds), ile sekund ma trwać jedna sesja

-v (verbose), czy ma się włączyć okienko z naszą twarzą

-a (audio), czy ma się włączać powiadomienie głosowe przy odpowiednio długim zamknięciu oczu

-c (closed_eyes_seconds_threshold), po jakim czasie zamknięte oczy liczą się jako sen

-f (FPS), ilość klatek na sekundę

-e (eye aspect ratio), próg EAR poniżej którego przymknięte oczy liczą się jako sen

-y (yawn_threshold), próg otwarcia ust powyżej którego wykrywane jest ziewanie

-t (time_limit), czy program ma działać aż go manualnie nie wyłączymy

-f (filepath), ścieżka do pliku do którego można zapisać statystyki z używania programu

Zachęcam do samodzielnego wypróbowania programu.