Pemanfaatan Sistem Face Tracking Berbasis CNN untuk Mendeteksi Microsleep pada Pengemudi Alat Berat di Lingkungan Pertambangan

Rifqi Naufal Abdjul - 13520062[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520062@itb.ac.id

***Abstract*—Makalah ini membahas tentang pengembangan sistem deteksi *microsleep* pada pengemudi alat berat di lingkungan pertambangan menggunakan teknologi *face tracking* berbasis *Convolutional Neural Networks* (CNN). Sistem ini bertujuan untuk mengurangi risiko kecelakaan akibat kelelahan pengemudi. Metode ini menganalisis fitur wajah dan memberikan peringatan dini apabila terdeteksi tanda-tanda kelelahan, seperti mata yang tertutup atau kedipan yang tidak normal. Penggunaan teknologi ini diharapkan dapat meningkatkan keselamatan kerja di sektor pertambangan dengan meminimalkan kesalahan manusia yang disebabkan oleh kelelahan.**

Keywords— computer vision, microsleep, facemesh

I. PENDAHULUAN

Batu bara merupakan sumber energi utama dalam produksi listrik di Indonesia. Data terkini menunjukkan bahwa hampir separuh dari listrik negara ini dihasilkan oleh pembangkit listrik tenaga uap yang menggunakan batu bara. Ini menandakan bahwa tambang batu bara memainkan peran vital dalam menyokong kebutuhan energi listrik di Indonesia. Kegiatan pertambangan batu bara melibatkan penggunaan alat berat seperti *excavator*, truk *loader*, dan *hauler* yang beroperasi secara terus-menerus. Pekerja tambang harus berkonsentrasi dan fokus penuh selama jam kerja mereka, yang dapat menyebabkan kelelahan dan distraksi. Kondisi ini seringkali menjadi faktor utama kecelakaan yang terjadi di area tambang batu bara. Untuk mengurangi risiko ini, sangat penting untuk memiliki sistem yang dapat meminimalisir kesalahan manusia di lingkungan pertambangan.

Pengemudi alat berat sering bekerja dalam shift yang panjang dan di bawah kondisi yang menuntut secara fisik dan mental. Hal ini meningkatkan risiko kelelahan, yang kemudian meningkatkan kemungkinan terjadinya microsleep. Mengidentifikasi dan mencegah *microsleep* sebelum terjadi adalah strategi yang jauh lebih efektif dan aman daripada menangani konsekuensinya. Sistem pendeteksi *microsleep* dapat memberikan peringatan dini kepada pengemudi yang menunjukkan tanda-tanda kelelahan, memungkinkan mereka untuk mengambil istirahat sebelum terjadi kecelakaan.

Penggunaan teknologi *computer vision* berbasis *Convolutional Neural Networks* (CNN) merupakan solusi inovatif untuk mendeteksi kelelahan pada wajah pengendara, khususnya dalam operasi alat berat. Dengan kemampuan CNN dalam mengenali pola visual, sistem ini secara efektif mengidentifikasi tanda-tanda kelelahan seperti mata yang tertutup atau kedipan yang tidak normal, serta perubahan ekspresi wajah. Keunggulan utamanya adalah kemampuan pemrosesan gambar secara *real-time*, yang memungkinkan deteksi kelelahan secara instan dan akurat. Selain itu, sistem ini dapat beradaptasi dengan berbagai kondisi pencahayaan, meningkatkan keandalannya dalam berbagai lingkungan kerja. Integrasi dengan sistem keselamatan lainnya, seperti peringatan suara atau kontrol otomatis, meningkatkan efektivitasnya dalam mencegah kecelakaan akibat kelelahan, menjadikan teknologi ini krusial untuk meningkatkan keselamatan dalam operasi pertambangan dan industri serupa.

II. TEORI DASAR

A. *Microsleep*

Gambar 2.1 Contoh indikator *microsleep*
Sumber: Diadaptasi dari [1]

Microsleep adalah fenomena singkat di mana seseorang mengalami episode tidur yang berlangsung beberapa detik. Biasanya terjadi ketika seseorang sangat lelah, tetapi tetap mencoba tetap terjaga. Selama *microsleep*, aktivitas otak menunjukkan pola yang mirip dengan tidur, meskipun orang

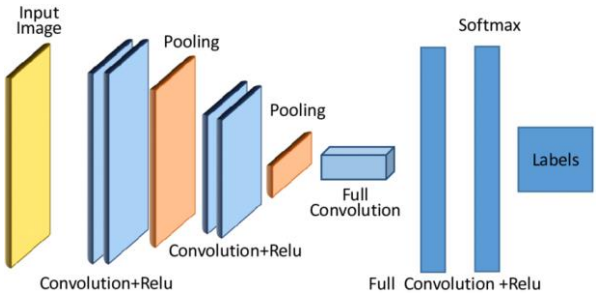
tersebut mungkin tampak terjaga.

Dari perspektif fitur wajah, *microsleep* bisa dikenali melalui beberapa indikator, seperti:

1. Penutupan mata, mata mungkin mulai tertutup secara perlahan atau berkedip lebih lambat dari biasanya. Ini adalah salah satu tanda paling umum dari *microsleep*.
2. Relaksasi otot wajah, selama *microsleep*, otot-otot wajah dapat relaksasi, menyebabkan ekspresi wajah tampak datar atau kosong. Ini karena pengurangan aktivitas saraf yang mengendalikan ekspresi wajah.
3. Kepala menjatuhkan, kepala mungkin mulai menunduk karena kehilangan kontrol otot leher. Ini sering terjadi ketika seseorang duduk dan berusaha tetap terjaga.
4. Gerakan mata cepat atau *Rapid Eye Movement* (REM), meski jarang, *microsleep* yang lebih dalam dapat menyertakan gerakan mata cepat yang mirip dengan yang terjadi selama tidur REM.
5. Perubahan Ekspresi: Mungkin ada perubahan singkat dalam ekspresi wajah yang menunjukkan kebingungan atau desorientasi saat seseorang "bangun" dari episode *microsleep*.

B. *Convolution Neural Network*

Convolutional Neural Network (CNN atau ConvNet) adalah arsitektur jaringan syaraf tiruan yang populer dalam pembelajaran mendalam (*deep learning*). CNN dirancang untuk memproses data yang memiliki topologi seperti grid, contohnya citra. CNN belajar langsung dari data, menghilangkan kebutuhan untuk melakukan ekstraksi fitur secara manual. Ini memungkinkan jaringan untuk mengidentifikasi fitur penting secara otomatis dari data yang diberikan.

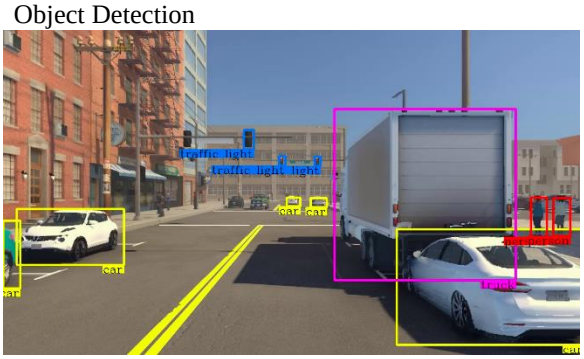


Gambar 2.2 Contoh arsitektur CNN
Sumber: Diadaptasi dari [2]

CNN merupakan kombinasi dari *Artificial Neural Network* (ANN) dengan operasi konvolusi. Konvolusi membantu dalam mengidentifikasi fitur penting dari data input. CNN terdiri dari tiga lapisan utama, yaitu:

1. *Convolutional Layer*
   Lapisan ini merupakan inti dari CNN. Di lapisan ini, operasi konvolusi dilakukan pada input menggunakan sekumpulan filter atau kernel. Filter tersebut bergeser melintasi input untuk menghasilkan *feature map*. Proses ini membantu dalam menangkap fitur-fitur penting seperti tepi, garis, dan tekstur dalam citra. Setelah proses konvolusi, biasanya diikuti oleh fungsi aktivasi seperti *Rectified Linear Unit* (ReLU). Fungsi aktivasi ini menambahkan non-linearitas ke dalam model, memungkinkan jaringan untuk menangkap kompleksitas dalam data. Lapisan konvolusi membantu dalam mendeteksi fitur lokal dari input, sehingga membuat model lebih efisien dan efektif dalam mempelajari fitur spesifik dari citra.
2. *Pooling Layer*
   Setelah *convolutional layer*, seringkali diikuti oleh *pooling layer*. Tujuan utama dari lapisan ini adalah untuk mengurangi dimensi spasial dari *feature map* yang dihasilkan oleh lapisan konvolusi. Ini dilakukan dengan merangkum atau mengambil nilai maksimum dari area-area tertentu pada *feature map*. Proses ini disebut sebagai *Max Pooling*. Ada juga variasi lain seperti *Average Pooling*. *Pooling* membantu dalam mengurangi jumlah parameter dan komputasi yang diperlukan oleh jaringan, juga mengurangi resiko *overfitting* dengan menyediakan bentuk abstraksi spasial. Selain itu, *pooling layer* membuat representasi output menjadi lebih invarian terhadap perubahan posisi kecil pada input.
3. *Fully-Connected Layer*
   Lapisan ini mirip dengan lapisan yang ada pada *Artificial Neural Network* (ANN) tradisional. Setelah melalui satu atau lebih *convolutional* dan *pooling layers*, data diproses melalui satu atau lebih *fully-connected layers* untuk klasifikasi akhir. Di lapisan ini, setiap *neuron* terhubung ke semua *neuron* di lapisan sebelumnya. Tujuan utama dari *fully-connected layer* adalah untuk menggunakan fitur-fitur yang telah diekstrak oleh *convolutional* dan *pooling layers* untuk melakukan klasifikasi atau prediksi. Pada dasarnya, lapisan ini bertindak sebagai 'pemikir' yang menginterpretasikan fitur-fitur yang telah diekstrak sebelumnya dan membuat keputusan berdasarkan informasi tersebut.

C. Object Detection



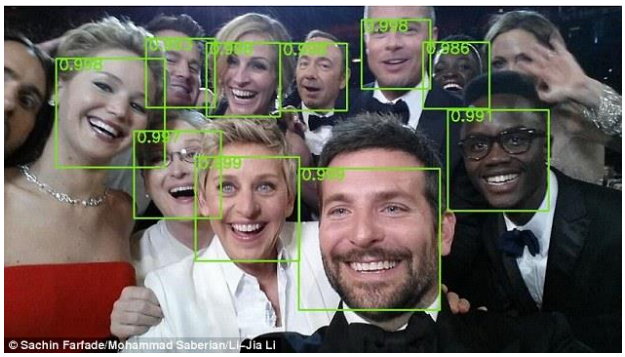
Gambar 2.3 Contoh hasil *object detection*
Sumber: Deploying a Scalable Object Detection Pipeline: The Inferencing Process, NVIDIA blog

Deteksi objek adalah salah satu aplikasi utama dari pembelajaran mendalam dalam bidang *computer vision*. Dalam konteks ini, deteksi objek merujuk pada kemampuan untuk mengidentifikasi dan melokalisasi objek dalam citra digital. Deteksi objek tidak hanya melibatkan klasifikasi (menentukan apa objek tersebut) tetapi juga lokalisasi (menemukan di mana

objek tersebut berada dalam citra). Ini berarti algoritma deteksi objek harus mampu memberikan label kelas kepada objek dan juga menentukan lokasi spesifik objek tersebut dalam citra, biasanya dengan menggunakan *bounding box*. Proses pelatihan model deteksi objek memerlukan daya komputasi yang tinggi, terutama karena kompleksitas algoritma dan ukuran dataset. Penggunaan GPU (*Graphics Processing Unit*) telah mengubah cara pembelajaran mendalam dilakukan, mempercepat proses pelatihan dari yang biasanya memakan waktu berminggu-minggu menjadi hanya beberapa jam.

Deteksi objek memiliki berbagai aplikasi praktis seperti dalam pengenalan wajah, pengawasan video, kendaraan otonom, dan banyak lagi. Teknologi ini memungkinkan mesin untuk mengenali objek dalam citra secara real-time, yang sangat berguna dalam aplikasi yang memerlukan respon cepat dan akurat. Salah satunya adalah deteksi keberadaan muka dari citra masukan.

D. Face Detection

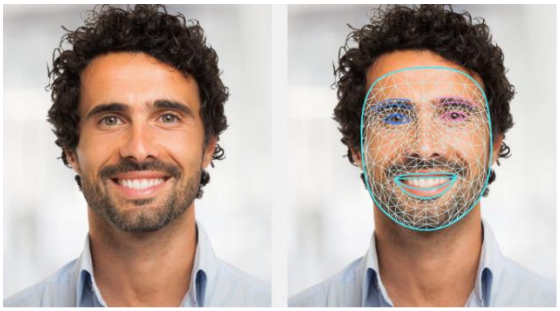


Gambar 2.4 Contoh hasil *face detection*
Sumber: Facial recognition breakthrough: 'Deep Dense' software spots faces in images even if they're partially hidden or UPSIDE DOWN, www.dailymail.co.uk

Deteksi wajah menggunakan teknologi deteksi objek dengan *Convolutional Neural Networks* (CNN) melibatkan identifikasi dan lokalisasi wajah dalam citra atau video. Salah satu metode yang digunakan dalam deteksi wajah adalah dengan memanfaatkan warna kulit. Warna kulit dapat dijadikan sebagai deskriptor yang berdayaguna dalam mengidentifikasi dan mengekstrak wajah dari citra. Hal ini didasarkan pada karakteristik warna kulit yang cenderung konsisten dan berbeda dari objek lain dalam citra. Dalam penggunaan CNN, jaringan belajar untuk mengenali fitur spesifik wajah, seperti bentuk mata, hidung, mulut, dan kontur wajah. Hal ini memungkinkan model untuk mengidentifikasi wajah dalam berbagai kondisi pencahayaan dan sudut pandang. CNN memiliki kemampuan untuk melakukan ekstraksi fitur secara otomatis melalui lapisan-lapisan konvolusi dan *pooling*. Ini memungkinkan jaringan untuk menangkap detail-detail penting dari wajah dan membedakannya dari latar belakang atau objek lain. Dalam banyak kasus, model CNN yang telah dilatih sebelumnya dengan dataset besar seperti ImageNet dapat digunakan sebagai titik awal. Model-model ini kemudian dapat disesuaikan atau dilatih lebih lanjut dengan dataset yang berfokus pada deteksi wajah untuk meningkatkan akurasi dan efisiensi dalam mengenali wajah manusia.

E. *Face Landmark* Mediapipe

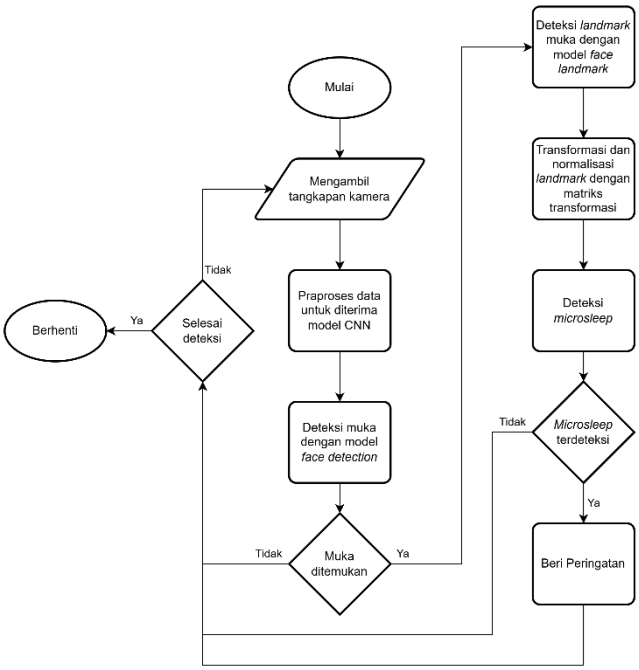
MediaPipe menyediakan model-model yang sudah dilatih sebelumnya untuk deteksi landmark wajah. Model-model ini telah dilatih pada dataset besar dan mampu mengidentifikasi berbagai titik penting pada wajah, seperti sudut mata, ujung hidung, bibir, dan lain-lain. Model CNN yang digunakan dalam MediaPipe dirancang untuk mengenali dan mengekstraksi fitur wajah secara akurat. CNN mampu menangkap detail wajah dengan melalui berbagai lapisan konvolusi dan pooling, sehingga dapat mengidentifikasi landmark wajah dengan presisi tinggi. Setelah deteksi landmark awal, MediaPipe dapat melakukan tracking terhadap landmark tersebut dalam video atau streaming citra secara real-time. Hal ini memungkinkan aplikasi untuk mengikuti pergerakan wajah dan ekspresi pengguna secara dinamis.



Gambar 2.5 Contoh hasil *face landmark* pustaka Mediapipe
Sumber: Diadaptasi dari [3]

III. IMPLEMENTASI

A. Gambaran Umum Sistem



Gambar 3.1 Alur umum sistem

Sistem dibangun dengan dasar teknologi *face tracking* yang

didukung oleh pustaka Mediapipe. Proses dimulai dengan mengambil tangkapan kamera yang kemudian akan dianalisis untuk mendeteksi landmark wajah menggunakan model face landmark. Setelah landmark terdeteksi, dilakukan transformasi dan normalisasi landmark dengan matriks transformasi untuk mempersiapkan data yang akan diproses oleh model CNN. Jika wajah berhasil ditemukan oleh model *face detection*, sistem akan melanjutkan untuk mendeteksi *microsleep*. Apabila *microsleep* terdeteksi, sistem akan memberikan peringatan kepada pengemudi. Jika tidak ada *microsleep* yang terdeteksi, proses deteksi dianggap selesai. Proses ini berlangsung secara berkelanjutan hingga diberhentikan, memastikan monitoring yang konstan terhadap pengemudi untuk mencegah kecelakaan akibat microsleep saat mengoperasikan alat berat di lingkungan pertambangan.

B. Sistem Pendeteksi *Face Landmark*

Implementasi sistem pendeteksi menggunakan program Python yang menggunakan pustaka OpenCV dan MediaPipe untuk mendeteksi *landmark* wajah dari input video secara *real-time*. Awalnya, sistem menginisialisasi kamera dan pembacaan frame dilakukan dalam loop. Dengan target 30 *frame* per detik, setiap frame diubah ke format warna yang sesuai dan diolah menggunakan MediaPipe untuk mendeteksi *landmark* wajah. Hasil deteksi ditampilkan pada jendela pengguna. Jika deteksi berhasil, proses tambahan dilakukan melalui fungsi process_*landmark*. Loop dapat dihentikan melalui input pengguna, dan setelah berakhir, sumber daya kamera dibebaskan dan semua jendela yang dibuka oleh OpenCV ditutup untuk mengakhiri sesi.

Untuk melakukan penormalisasian hasil deteksi, dapat digunakan matriks transformasi yang dikeluarkan dari model Mediapipe. Seluruh titik hasil deteksi akan ditransformasikan dengan matriks transformasi ke dalam ukuran 0-1 agar tidak terpengaruh dari jarak dan arah tangkapan kamera.

C. Sistem Pendeteksi *Microsleep*

Untuk melakukan deteksi *microsleep*, sistem menyimpan data *global* yang dapat diakses dalam fungsi manapun. Lalu sistem menggunakan jumlah kedipan, menguap, dan tanda kelelahan untuk mendeteksi *microsleep*. Ketiga hal tersebut dijelaskan sebagai berikut,

```
blink_counter = 0
yawn_timestamp = None

blinking = False
is_tired = False

eye_history = np.nan * np.ones(WINDOW_SIZE)
mouth_history = np.nan * np.ones(WINDOW_SIZE)
eye_history_index = 0
mouth_history_index = 0
blink_history = np.nan * np.ones(MIN_BLINK_COUNT)
blink_history_index = 0
```

1. Deteksi Kedipan

Sistem menyimpan riwayat dari *aspect ratio* mata (rasio antara lebar mata dengan tingginya). Jika *aspect ratio* saat ini turun di bawah ambang batas tertentu (menandakan mata lebih tertutup dari biasanya), ini dianggap sebagai kedipan. Ambang batas ini dihitung secara dinamis berdasarkan rata-rata dari *aspect ratio* dalam riwayat, dengan mengesampingkan 10% nilai teratas dan terbawah untuk menghindari data outlier. Berikut adalah kode python untuk mendeteksi kedipan.

```
def check_blink(timestamp):
    global eye_history, eye_history_index,
blink_counter, blinking, blink_history,
blink_history_index
    eye_threshold = (
        float(
            np.mean(
               np.sort(eye_history)[int(WINDOW_SIZE
* 0.1) : int(WINDOW_SIZE * 0.9)]
            )
        )
        * 1.2
    )
    if not np.isnan(eye_threshold) and
eye_history[eye_history_index] < eye_threshold:
        if not blinking:
            blink_counter += 1
            blink_history[blink_history_index] =
timestamp
            blink_history_index =
(blink_history_index + 1) % MIN_BLINK_COUNT
        blinking = True
    else:
        blinking = False
```

2. Deteksi Menguap

Sistem menghitung luas dari bounding box sekitar mulut. Jika luas ini melebihi ambang batas tertentu (menunjukkan mulut terbuka lebar), ini dianggap sebagai menguap. Berikut adalah kode python untuk mendeteksi menguap.

```
def check_yawn(timestamp):
    global mouth_history, mouth_history_index,
yawn_timestamp
    mouth_area =
mouth_history[mouth_history_index]
    if mouth_area > 0.03:
        yawn_timestamp = timestamp
```

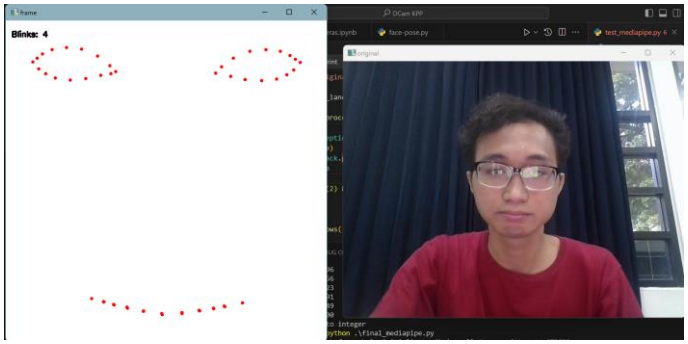
3. Deteksi Tanda Kelelahan

Sistem memeriksa dua kondisi untuk mendeteksi kelelahan. Pertama, script memeriksa apakah rentang dari aspect ratio mata dalam setengah terakhir dari riwayat adalah kurang dari 1, menunjukkan mata telah relatif stabil dan mungkin setengah tertutup. Kedua, script memeriksa apakah telah terjadi kurang dari 10 kedipan dalam waktu 5 menit terakhir. Jika salah satu kondisi terpenuhi, script akan menandai pengguna sebagai kelelahan. Berikut adalah kode *python* untuk mendeteksi kelelahan.

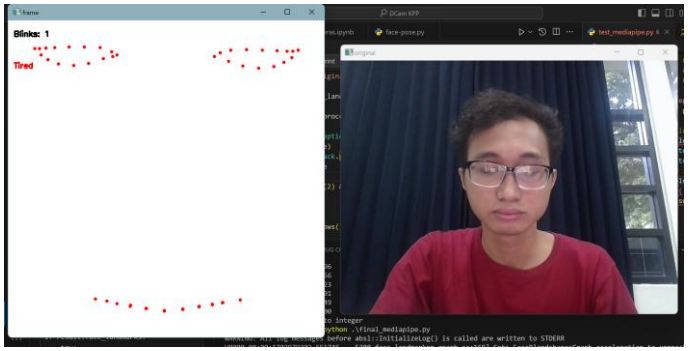
```
def check_tired(timestamp):
    global eye_history, eye_history_index,
is_tired, blink_history, blink_history_index,
blink_counter
    if np.isnan(np.mean(eye_history)):
        return
    # check last eye_history (half window size)
range is less than 1
    half_window_size = int(WINDOW_SIZE / 2)
    if eye_history_index < half_window_size:
        half_eye_history = np.concatenate(
            (
                eye_history[:eye_history_index],
                eye_history[-(half_window_size -
eye_history_index) :],
            )
        )
    else:
        half_eye_history = eye_history[
            eye_history_index - half_window_size :
eye_history_index
        ]
    if float(np.max(half_eye_history)) -
float(np.min(half_eye_history)) < 1:
        is_tired = True
    else:
        # For every blinkcounttimer, check blink
count and reset blink count, if less than min blink
count, is_tired = True
        if timestamp - np.nanmin(blink_history) >
BLINK_COUNT_TIMER * 1000:
            if blink_counter < MIN_BLINK_COUNT:
                is_tired = True
            else:
                is_tired = False
            blink_counter = 0
            blink_history = np.nan *
np.ones(MIN_BLINK_COUNT)
            blink_history_index = 0
```

D. Hasil Implementasi

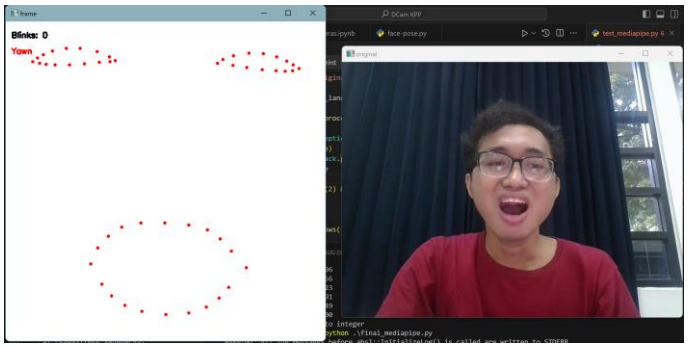
Hasil tangkapan layar ditampilkan dengan 2 gambar, gambar pertama merupakan hal yang dilihat oleh sistem berisikan hasil normalisasi *landmark* dan jumlah kedipan dan peringatan ketika menguap dan kelelahan. Berikut terdapat beberapa hasil tangkapan layar dari hasil implementasi.



Gambar 3.2 Gambaran normal dari sistem



Gambar 3.3 Gambaran sistem ketika terdapat kelelahan



Gambar 3.4 Gambaran sistem ketika menguap

IV. KESIMPULAN

Sistem pendeteksi *microsleep* dengan tanda-tandanya seperti kurangnya mengedip, mata tidak resnposif, dan menguap berhasil di implementasi dengan performa yang cukup baik. Dengan memanfaatkan teknologi *machine learning* yang dapat diakses dengan mudah di internet, kita dapat membuat sistem yang dapat membantu menyelamatkan nyawa.

VII. UCAPAN TERIMA KASIH

Penulis ingin mengucap syukur kepada Allah Swt. karena atas berkat yang diberikan-Nya sehingga penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga ingin berterima kasih kepada pak Rinaldi Munir selaku dosen mata kuliah Interpretasi dan Pengolahan Citra IF4073 yang telah memberikan materi dan pengetahuan untuk menyusun makalah ini. Tidak lupa juga teman yang memberikan dukungan selama proses penulisan makalah ini.

REFERENSI

[1] Kir Savaş, Burcu & Becerkli, Yaşar. (2020). Real Time Driver Fatigue Detection System Based on Multi-Task ConNN. IEEE Access. 8. 1-1. 10.1109/ACCESS.2020.2963960.

[2] Afridi, Tariq & Alam, Aftab & Khan, Numan & Khan, Jawad. (2020). A Multimodal Memes Classification: A Survey and Open Research Issues.

[3] "MediaPipe | Google for Developers," Google for Developers. https://developers.google.com/mediapipe/solutions

[4] R. Munir, "21-CNN-2023," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/21-CNN-2023.pdf. Diakses pada 16 Desember 2023.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 16 Desember 2023

Rifqi Naufal Abdjul
13520062