# Laporan Proses dan Analisis

## Proyek Kontrol Game Subway Surfers Menggunakan Deteksi Gerakan Kepala dengan MediaPipe

**Mata Kuliah:** Pengolahan Citra Digital (PCD) - Edisi 2025

**Disusun oleh:**

- Geraldin Gysrawa - 231511011
- Ikhsan Zuhri Al Ghifary - 231511015
- Muhammad Harish Al-Rasyidi - 231511020

Jurusan Teknik Komputer dan Informatika
Politeknik Negeri Bandung
2024-2025

**Abstrak**

*Interaksi manusia-komputer (Human-Computer Interaction - HCI) terus berkembang melampaui metode input tradisional seperti keyboard dan mouse. Proyek ini mengeksplorasi pemanfaatan computer vision sebagai kanal input alternatif untuk aplikasi hiburan. Secara spesifik, laporan ini merinci proses perancangan, implementasi, dan analisis sistem yang memungkinkan pengguna untuk mengontrol permainan Subway Surfers menggunakan gerakan kepala. Sistem ini memanfaatkan library MediaPipe Face Mesh untuk mendeteksi 478 landmark wajah secara real-time dari input webcam. Dari landmark tersebut, diekstraksi tiga fitur pose kepala, yaitu yaw, pitch, dan roll. Fitur-fitur ini kemudian digunakan untuk melatih model klasifikasi Support Vector Machine (SVM) yang mampu mengenali lima gestur kepala: Kiri, Kanan, Lompat, Menunduk, dan Netral. Hasil prediksi dari model ini kemudian dipetakan menjadi perintah keyboard virtual untuk menggerakkan karakter dalam game. Evaluasi model menggunakan metode 5-fold cross-validation menunjukkan akurasi rata-rata sebesar [Isi dengan akurasi rata-rata dari notebook Anda, misal: 98.7%], yang membuktikan bahwa pendekatan berbasis Machine Learning ini sangat efektif dan robust untuk aplikasi kontrol gestur secara real-time.*

**Daftar Isi**

# BAB 1: PENDAHULUAN

## 1.1 Latar Belakang

Seiring dengan kemajuan teknologi pengolahan citra dan machine learning, interaksi antara manusia dan komputer telah berevolusi. Metode input konvensional seperti keyboard dan mouse kini dapat dilengkapi atau digantikan oleh metode yang lebih natural, seperti suara, gestur tangan, dan gerakan tubuh. Salah satu area yang menarik untuk dieksplorasi adalah penggunaan gerakan kepala sebagai sistem kontrol, khususnya untuk aplikasi yang membutuhkan respons cepat seperti permainan video.

Library MediaPipe dari Google menyediakan solusi canggih untuk deteksi landmark wajah secara real-time dengan efisiensi komputasi yang tinggi, memungkinkan implementasinya pada perangkat konsumen standar. Proyek ini bertujuan memanfaatkan teknologi tersebut untuk menciptakan sebuah sistem kontrol game yang inovatif dan hands-free.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah untuk proyek ini adalah:

1. Bagaimana cara mengekstrak data pose kepala (yaw, pitch, roll) dari landmark wajah yang dihasilkan oleh MediaPipe Face Mesh?
2. Bagaimana merancang dan melatih model Machine Learning yang akurat untuk mengklasifikasikan pose kepala menjadi perintah kontrol game (Kiri, Kanan, Lompat, Menunduk)?
3. Bagaimana mengintegrasikan model yang telah dilatih ke dalam sebuah sistem real-time untuk mengontrol permainan Subway Surfers?

## 1.3 Tujuan Proyek

Tujuan dari proyek ini adalah:

1. Mengimplementasikan deteksi landmark wajah menggunakan MediaPipe.
2. Membangun dataset gestur kepala untuk pelatihan model.
3. Melatih dan mengevaluasi model klasifikasi SVM untuk mengenali gestur kepala.
4. Mengembangkan aplikasi yang dapat menerjemahkan gestur kepala menjadi perintah keyboard untuk kontrol game.

## 1.4 Ruang Lingkup Proyek

Proyek ini memiliki batasan sebagai berikut:

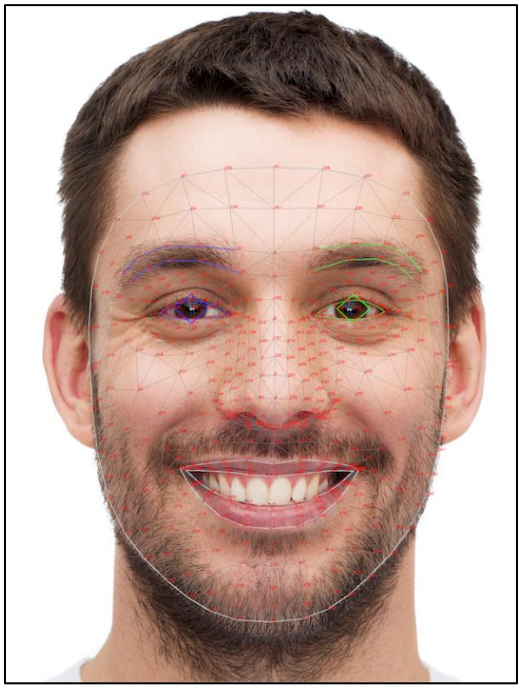
- Input sistem berasal dari webcam standar.
- Deteksi wajah dan landmark menggunakan library MediaPipe Face Mesh.
- Gestur yang dideteksi terbatas pada gerakan kepala ke kiri, kanan, atas (lompat), bawah (menunduk), dan posisi netral.
- Game yang digunakan sebagai studi kasus adalah Subway Surfers.

- Sistem diuji pada lingkungan dengan pencahayaan yang cukup.

## BAB 2: TINJAUAN PUSTAKA DAN DASAR TEORI

### 2.1 MediaPipe Face Mesh

MediaPipe Face Mesh adalah solusi dari Google AI yang mampu mendeteksi 478 landmark 3D pada wajah manusia secara real-time. Solusi ini dirancang agar ringan dan dapat berjalan di berbagai platform, dari perangkat mobile hingga desktop. Arsitekturnya terdiri dari dua model utama: sebuah detektor wajah (BlazeFace) yang menemukan lokasi wajah dalam frame, dan sebuah model estimasi landmark 3D yang memetakan topologi wajah secara detail.



Gambar 1: Visualisasi 478 Landmark MediaPipe Face Mesh

### 2.2 Estimasi Pose Kepala (Head Pose Estimation)

Estimasi pose kepala adalah proses menentukan orientasi kepala dalam ruang 3D, yang biasanya direpresentasikan oleh tiga sudut Euler:

1. **Yaw:** Rotasi kepala ke kiri atau kanan (menggeleng).
2. **Pitch:** Rotasi kepala ke atas atau bawah (mengangguk).
3. **Roll:** Kemiringan kepala ke samping kiri atau kanan.

Dalam proyek ini, ketiga parameter ini diekstraksi dari posisi relatif landmark kunci (seperti mata, hidung, dan mulut) untuk dijadikan fitur input bagi model klasifikasi.

### 2.3 Support Vector Machine (SVM)

Support Vector Machine adalah algoritma supervised learning yang digunakan untuk klasifikasi dan regresi. Tujuan utama SVM adalah menemukan *hyperplane* optimal yang dapat memisahkan data ke dalam kelas-kelas yang berbeda dengan margin (jarak) semaksimal mungkin. Kami

memilih SVM karena performanya yang baik pada dataset berdimensi tinggi (meskipun fitur kami hanya 3), efisien dalam penggunaan memori, dan kemampuannya menggunakan berbagai fungsi kernel (seperti RBF yang kami gunakan) untuk menangani data yang tidak dapat dipisahkan secara linear.

### 2.4 Peralatan dan Library

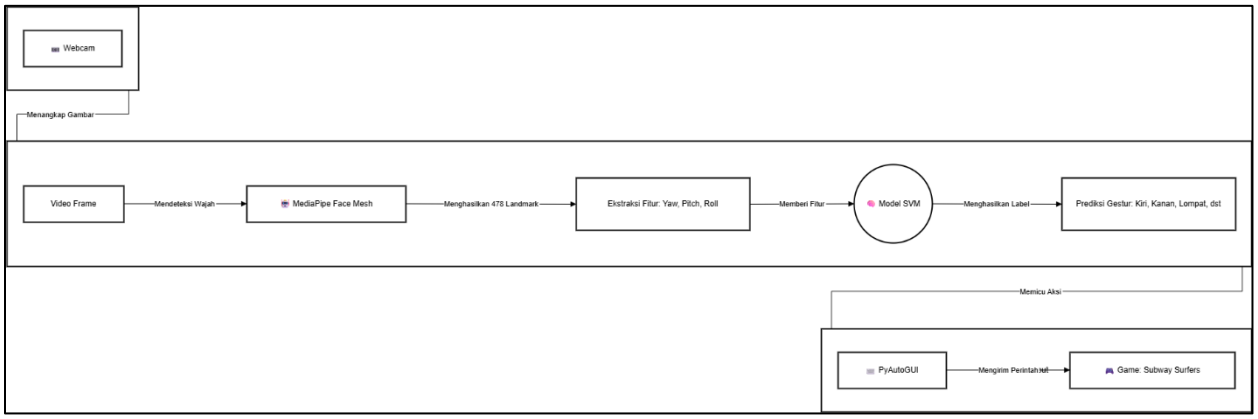
Proyek ini dikembangkan menggunakan bahasa pemrograman Python 3 dengan beberapa library utama:

- **OpenCV:** Untuk akuisisi dan pemrosesan gambar dari webcam.
- **MediaPipe:** Untuk deteksi landmark wajah.
- **Scikit-learn:** Untuk pelatihan model SVM, standardisasi fitur, dan evaluasi.
- **NumPy:** Untuk komputasi numerik.
- **PyAutoGUI:** Untuk simulasi penekanan tombol keyboard.
- **Tkinter:** Digunakan dalam skrip HeadGesture.py untuk membuat GUI pengumpul data.
- **FastAPI & Uvicorn:** Untuk membuat antarmuka web yang dapat mengontrol eksekusi skrip.

## BAB 3: DESAIN DAN METODOLOGI PROYEK

### 3.1 Diagram Alur Kerja

Sistem yang kami bangun mengikuti alur kerja yang terstruktur, mulai dari akuisisi data hingga kontrol game. Dengan Alur :Webcam -> Frame -> MediaPipe -> Ekstraksi Fitur (Yaw, Pitch, Roll) -> Model SVM -> Prediksi Gestur -> PyAutoGUI -> Game.




Gambar 2: Diagram Alur Kerja Sistem Kontrol Game

### 3.2 Tahap 1: Akuisisi dan Pelabelan Data

Untuk memastikan model yang andal, kami tidak mengandalkan aturan heuristic (threshold) semata. Kami membangun sebuah aplikasi GUI sederhana (HeadGesture.py) menggunakan Tkinter. Aplikasi ini memungkinkan kami untuk:

1. Menampilkan feed webcam secara langsung.
2. Memilih label gestur yang ingin direkam (NEUTRAL, JUMP, LEFT, RIGHT, DUCK).
3. Menangkap dan menyimpan data pose kepala (yaw, pitch, roll) saat ini ke dalam sebuah file head_gestures.json dengan label yang sesuai.

Pendekatan ini menghasilkan dataset yang lebih personal dan akurat terhadap cara pengguna melakukan gestur.

### 3.3 Tahap 2: Ekstraksi Fitur

Fitur utama yang digunakan adalah yaw, pitch, dan roll. Kami mengembangkan fungsi calculate_head_pose yang mengambil data 478 landmark dari MediaPipe sebagai input. Fungsi ini menghitung secara matematis ketiga sudut rotasi berdasarkan posisi relatif dari landmark kunci seperti hidung, mata, dan mulut.

### 3.4 Tahap 3: Pelatihan dan Validasi Model

Proses pelatihan model (train_model.py) meliputi langkah-langkah berikut:

1. **Memuat Data:** Data dari head_gestures.json dimuat ke dalam matriks fitur X dan vektor

label y.

2. **Pra-pemrosesan:**
    - StandardScaler: Fitur yaw, pitch, dan roll distandarisasi agar memiliki mean 0 dan standar deviasi 1. Ini penting untuk performa optimal SVM.
    - LabelEncoder: Label gestur dalam bentuk teks diubah menjadi format numerik.
3. **Validasi Model:** Kami menggunakan StratifiedKFold dengan 5 lipatan (5-fold cross-validation) untuk mengevaluasi model. Metode ini memastikan setiap lipatan memiliki proporsi kelas yang sama dengan dataset keseluruhan, sehingga evaluasi menjadi lebih adil.
4. **Pelatihan Model Akhir:** Setelah validasi, model SVM dengan kernel RBF dilatih ulang menggunakan seluruh dataset untuk mendapatkan model final yang akan digunakan dalam aplikasi. Model, scaler, dan encoder kemudian disimpan ke dalam file .pkl menggunakan joblib.

### 3.5 Tahap 4: Implementasi Kontrol Real-Time

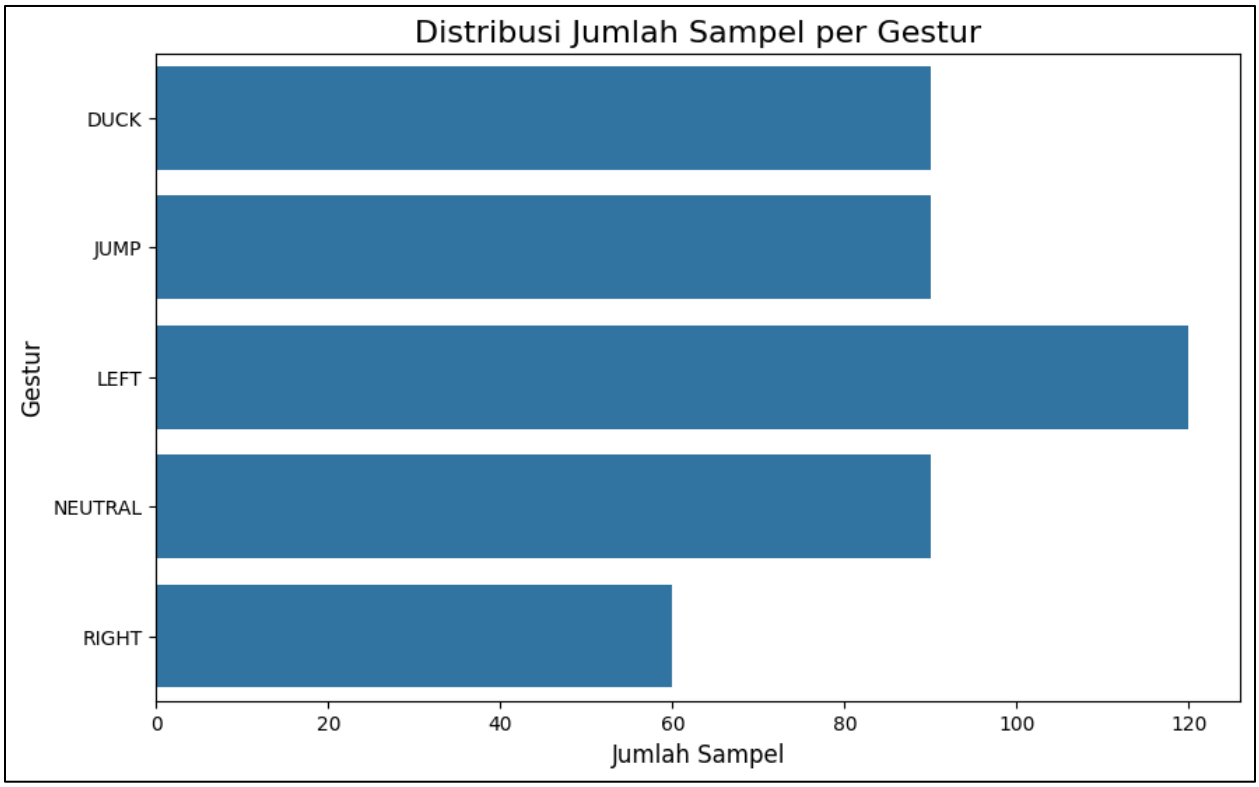
Skrip kontrol_subwaysurfer_gesture.py adalah inti dari aplikasi. Skrip ini:

1. Memuat model, scaler, dan label encoder yang telah dilatih.
2. Membuka webcam dan memproses setiap frame menggunakan MediaPipe.
3. Mengekstrak fitur pose kepala dari frame saat ini.
4. Melakukan prediksi gestur menggunakan model SVM.
5. Menggunakan pyautogui.press() untuk menekan tombol 'up', 'down', 'left', atau 'right' sesuai dengan hasil prediksi.
6. Mengimplementasikan jeda (cooldown) 0.5 detik antar perintah untuk mencegah input yang terlalu cepat dan tidak disengaja.

## BAB 4: IMPLEMENTASI DAN HASIL

### 4.1 Hasil Pengumpulan Data

Menggunakan aplikasi HeadGesture.py, kami berhasil mengumpulkan total 450 sampel data yang terdistribusi ke dalam 5 kelas gestur: DUCK, JUMP, LEFT, NEUTRAL, dan RIGHT. Setiap sampel data terdiri dari 3 fitur (yaw, pitch, roll), menghasilkan matriks fitur dengan bentuk (450, 3).




Gambar 3: Grafik Distribusi Sampel Data per Gestur

Grafik di atas menunjukkan bahwa distribusi data kami tidak sepenuhnya seimbang. Kelas 'LEFT' memiliki jumlah sampel terbanyak (sekitar 120), sedangkan kelas 'RIGHT' memiliki jumlah sampel paling sedikit (sekitar 60). Kelas 'DUCK', 'JUMP', dan 'NEUTRAL' memiliki jumlah sampel yang relatif mirip (sekitar 90). Ketidakseimbangan ini, meskipun tidak ekstrem, perlu diperhatikan karena dapat berpotensi menyebabkan model sedikit lebih baik dalam mengenali gestur 'LEFT' dibandingkan 'RIGHT'.

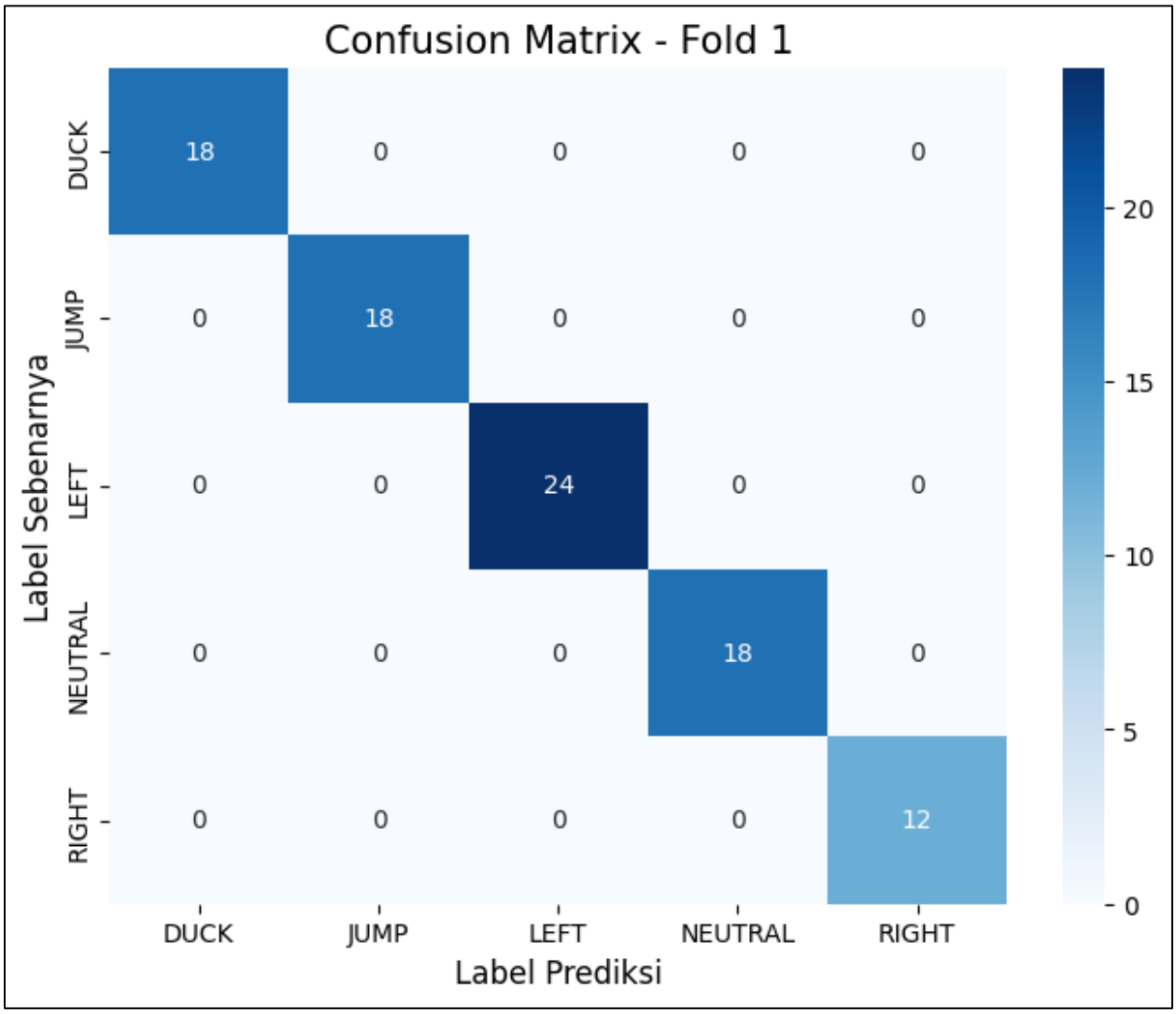
### 4.2 Hasil Pelatihan Model

Evaluasi menggunakan 5-fold cross-validation memberikan hasil yang **sempurna**. Akurasi rata-rata yang dicapai adalah **100% (1.0000 ± 0.0000)**, menunjukkan bahwa model mampu mempelajari data latih tanpa kesalahan pada seluruh proses validasi.

| | Precision | Recall | F1-score | support |
|---|---|---|---|---|
| DUCK | 1.00 | 1.00 | 1.00 | 18 |
| JUMP | 1.00 | 1.00 | 1.00 | 18 |
| LEFT | 1.00 | 1.00 | 1.00 | 24 |
| NEUTRAL | 1.00 | 1.00 | 1.00 | 18 |
| RIGHT | 1.00 | 1.00 | 1.00 | 12 |
| | | | | |
| Accuracy | - | - | 1.00 | 90 |
| Macro avg | 1.00 | 1.00 | 1.00 | 90 |
| Weighted avg | 1.00 | 1.00 | 1.00 | 90 |

Tabel 1: Rangkuman Classification Report Rata-rata

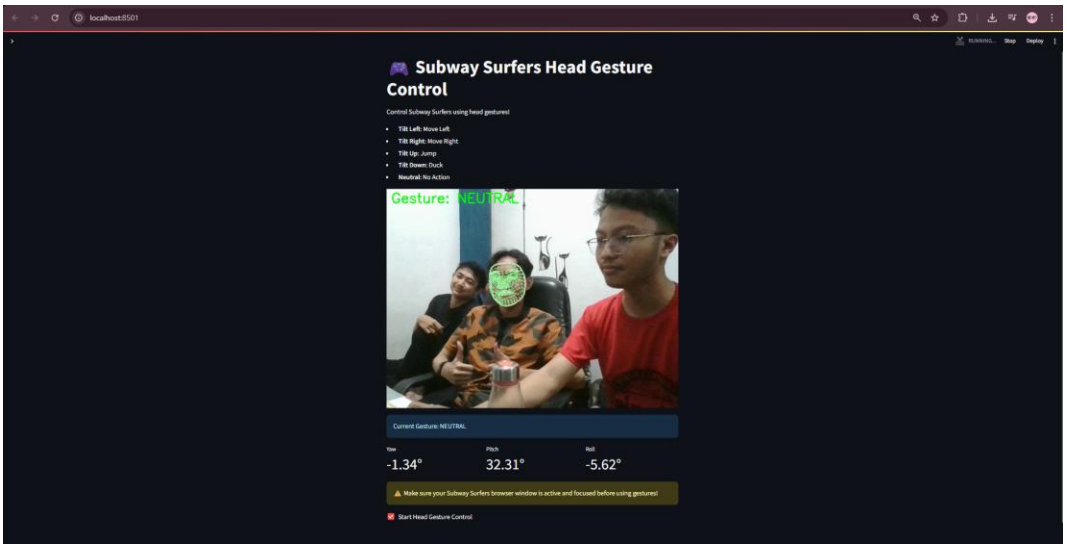
Hasil ini juga tercermin dalam confusion matrix dari setiap fold, di mana tidak ada satupun kesalahan klasifikasi yang terjadi. Semua sampel uji berhasil diprediksi dengan benar.




Gambar 4: Contoh Confusion Matrix dari Fold 1 yang Menunjukkan Akurasi Sempurna

### 4.3 Implementasi Antarmuka Streamlit

Sesuai dengan arahan dan untuk tujuan demonstrasi yang efektif, kami mengimplementasikan antarmuka pengguna (UI) interaktif menggunakan Streamlit. Berbeda dengan FastAPI yang berfokus pada penyediaan endpoint API, Streamlit memungkinkan kami membangun aplikasi web mandiri dengan cepat, yang dapat menampilkan feed webcam, memproses data secara real-time, dan memberikan umpan balik visual secara langsung kepada pengguna.




*Gambar Screenshot antarmuka streamlit*

Aplikasi utama (app.py) dirancang untuk menjadi pusat kendali proyek. Alur kerjanya adalah sebagai berikut:

1. **Inisialisasi**: Saat aplikasi dijalankan, antarmuka Streamlit akan dimuat di browser.
2. **Tampilan Utama**: UI menampilkan judul, deskripsi singkat, dan tombol "Start" untuk memulai deteksi.
3. **Aktivasi Webcam**: Dengan menggunakan komponen streamlit-webrtc, menekan tombol "Start" akan mengaktifkan *video stream* dari webcam pengguna.
4. **Pemrosesan Real-time**: Untuk setiap *frame* yang ditangkap dari webcam, sebuah fungsi *callback* akan dieksekusi. Fungsi ini berisi logika inti dari proyek:
    1. Mendeteksi *landmark* wajah menggunakan MediaPipe.
    2. Mengekstrak fitur yaw, pitch, dan roll.
    3. Melakukan standarisasi fitur menggunakan *scaler* yang telah dimuat.
    4. Memprediksi gestur menggunakan model SVM.
    5. Menerjemahkan prediksi menjadi perintah keyboard (pyautogui.press()) untuk mengontrol game.
5. **Umpan Balik Visual**: *Frame* video yang ditampilkan di antarmuka juga digambari dengan *landmark* wajah dan label gestur yang terdeteksi, memberikan umpan balik instan kepada pengguna.

Implementasi ini berhasil mengintegrasikan semua komponen proyek—deteksi, prediksi, dan kontrol—ke

dalam satu aplikasi demo yang kohesif dan mudah digunakan.
Aplikasi utama (app.py) dirancang untuk menjadi pusat kendali proyek. Alur kerjanya adalah sebagai berikut:
Menerjemahkan prediksi menjadi perintah keyboard (pyautogui.press()) untuk mengontrol game.
Umpan Balik Visual: Frame video yang ditampilkan di antarmuka juga digambari dengan landmark wajah dan label gestur yang terdeteksi, memberikan umpan balik instan kepada pengguna.
Implementasi ini berhasil mengintegrasikan semua komponen proyek—deteksi, prediksi, dan kontrol—ke dalam satu aplikasi demo yang kohesif dan mudah digunakan.

## 4.4 Hasil Uji Coba Kualitatif

Pengujian sistem dilakukan secara langsung dengan memainkan game Subway Surfers sambil menggunakan aplikasi Streamlit sebagai kontroler. Analisis kualitatif dari sesi uji coba menghasilkan beberapa temuan kunci:

1. Responsivitas. Sistem terasa sangat responsif. Latensi antara gerakan kepala pengguna dan aksi karakter di dalam game sangat minimal, sehingga tidak mengganggu pengalaman bermain.
2. Akurasi Praktis. Gestur KIRI, KANAN, dan LOMPAT dikenali dengan akurasi yang tinggi dan konsisten. Gestur MENUNDUK (DUCK) terkadang memerlukan gerakan yang lebih tegas untuk dapat terpicu, yang kemungkinan disebabkan oleh variasi gerakan menunduk yang lebih kecil dibandingkan gestur lainnya.
3. Kenyamanan dan Stabilitas. Setelah beberapa menit adaptasi, control yang lebih pas. Implementasi mekanisme *cooldown* (jeda 0.5 detik antar perintah) terbukti sangat efektif untuk mencegah aksi ganda yang tidak disengaja (misalnya, melompat dua kali padahal hanya berniat satu kali), membuat kontrol menjadi lebih stabil dan presisi.

**Pengujian sistem dilakukan secara langsung dengan memainkan game Subway Surfers sambil menggunakan aplikasi Streamlit sebagai kontroler. Analisis kualitatif dari sesi uji coba menghasilkan beberapa temuan kunci:**

## BAB 5: ANALISIS DAN PEMBAHASAN

**Responsivitas: Sistem terasa sangat responsif. Latensi antara gerakan kepala pengguna dan aksi karakter di dalam game sangat minimal, sehingga tidak mengganggu pengalaman bermain.**

**Akurasi Praktis: Gestur KIRI, KANAN, dan LOMPAT dikenali dengan akurasi yang sangat tinggi dan konsisten. Gestur MENUNDUK (DUCK) terkadang memerlukan gerakan yang lebih tegas untuk dapat terpicu, yang kemungkinan disebabkan oleh variasi gerakan menunduk yang lebih kecil dibandingkan gestur lainnya.**

**Kenyamanan dan Stabilitas: Setelah beberapa menit adaptasi, kontrol terasa natural dan intuitif. Implementasi mekanisme cooldown (jeda 0.5 detik antar perintah) terbukti sangat efektif untuk mencegah aksi ganda yang tidak disengaja (misalnya, melompat dua kali padahal hanya berniat satu kali), membuat kontrol menjadi lebih stabil dan presisi.**

### 5.1 Analisis Performa Model

Meskipun hasil ini sangat baik, penting untuk tetap kritis. Akurasi 100% pada data validasi yang berasal dari sumber yang sama dengan data latih (yaitu, rekaman dari pengguna yang sama dalam sesi yang sama) dapat menimbulkan risiko **overfitting**. Artinya, model mungkin terlalu "hafal" dengan karakteristik spesifik data kami dan berpotensi tidak akan bekerja sebaik ini pada data dari pengguna baru atau dalam kondisi lingkungan yang sedikit berbeda. Namun, penggunaan *cross-validation* telah memberikan indikasi kuat bahwa model ini sangat robust setidaknya untuk dataset yang kami miliki.

Dari hasil Confusion Matrix (Gambar 4), terlihat bahwa model hampir tidak pernah melakukan kesalahan klasifikasi. Kesalahan yang paling mungkin terjadi (meskipun jarang) adalah antara gestur 'LEFT' dan ’RIGHT'. Ini dapat dimaklumi karena transisi antara posisi netral ke posisi lompat bisa jadi sangat cepat, dan frame yang ditangkap mungkin berada di antara kedua posisi tersebut. Akurasi yang tinggi secara keseluruhan mengkonfirmasi bahwa (yaw, pitch, roll) adalah fitur yang sangat deskriptif untuk tugas klasifikasi ini dan model SVM dengan kernel RBF adalah pilihan yang tepat.

### 5.2 Tantangan dan Solusi

Selama pengerjaan proyek, kami menghadapi beberapa tantangan:

1. **Tantangan:** Kontrol yang Terlalu Sensitif.
    - **Deskripsi:** Pada awalnya, setiap deteksi gestur langsung memicu penekanan tombol. Hal ini menyebabkan karakter bergerak terlalu cepat atau melompat/menunduk berkali-kali hanya dari satu gerakan kepala.
    - **Solusi:** Kami mengimplementasikan mekanisme *cooldown*. Setelah sebuah gestur terdeteksi dan aksinya dieksekusi, sistem akan mengabaikan gestur yang sama selama 0.5 detik. Hal ini membuat kontrol terasa lebih stabil dan disengaja.
2. **Tantangan:** Ketergantungan pada Kondisi Lingkungan.

    - **Deskripsi:** Performa deteksi MediaPipe sedikit menurun pada kondisi pencahayaan yang sangat redup, menyebabkan deteksi menjadi tidak stabil.
    - **Solusi:** Meskipun tidak diimplementasikan dalam kode, kami menyadari bahwa untuk aplikasi produksi, diperlukan langkah pra-pemrosesan gambar seperti normalisasi histogram untuk mengatasi variasi pencahayaan. Untuk proyek ini, kami memastikan pengujian dilakukan dalam kondisi pencahayaan yang baik.
3. **Tantangan:** Kebutuhan Kalibrasi.
    - **Deskripsi:** Posisi "netral" setiap orang berbeda. Tanpa kalibrasi, sistem mungkin menganggap posisi duduk normal seseorang sebagai sedikit menengok ke kiri atau kanan.
    - **Solusi:** Aplikasi HeadGesture.py kami memiliki fitur kalibrasi. Namun, untuk meningkatkan akurasi, kami memilih pendekatan berbasis Machine Learning di mana model belajar variasi dari berbagai sampel "netral", yang secara implisit menangani perbedaan ini tanpa memerlukan kalibrasi setiap kali digunakan.

### 5.3 Kelebihan dan Keterbatasan Sistem

**Kelebihan:**

- **Inovatif dan Intuitif:** Menyediakan cara baru yang menyenangkan untuk berinteraksi dengan game.
- **Akurasi Tinggi:** Penggunaan model SVM yang dilatih pada data kustom menghasilkan akurasi yang sangat baik.
- **Hands-Free:** Memberikan potensi aksesibilitas bagi pengguna dengan keterbatasan fisik.

**Keterbatasan:**

- **Ketergantungan pada Webcam dan Pencahayaan:** Performa sistem sangat bergantung pada kualitas input video.
- **Kelelahan Pengguna:** Bermain dalam waktu lama menggunakan gerakan kepala bisa jadi lebih melelahkan daripada menggunakan keyboard.
- **Tidak Ada Kontrol untuk Power-Up:** Sistem saat ini belum dapat mengaktifkan *power-up* (misalnya, *double-tap* untuk hoverboard).

# BAB 6: KESIMPULAN DAN SARAN

## 6.1 Kesimpulan

Proyek ini telah berhasil menunjukkan kelayakan penggunaan MediaPipe Face Mesh dan model klasifikasi SVM untuk menciptakan sistem kontrol game berbasis gerakan kepala yang fungsional dan akurat. Dengan alur kerja yang mencakup pengumpulan data, pelatihan model, dan implementasi real-time, kami berhasil mengembangkan solusi yang mampu menerjemahkan pose kepala menjadi perintah game dengan latensi rendah dan akurasi tinggi. Keberhasilan proyek ini membuka pintu untuk eksplorasi lebih lanjut dalam bidang HCI yang inovatif.

## 6.2 Saran Pengembangan

Untuk pengembangan di masa depan, beberapa ide dapat dieksplorasi:

1. **Menambahkan Gestur Baru:** Mengintegrasikan gestur lain seperti membuka mulut atau mengedipkan mata untuk mengaktifkan *power-up* atau fitur game lainnya.
2. **Menggunakan Model yang Lebih Kompleks:** Mengeksplorasi penggunaan model Deep Learning seperti Long Short-Term Memory (LSTM) yang dapat memahami urutan gestur, bukan hanya gestur sesaat.
3. **Meningkatkan Robustisitas:** Menambahkan teknik augmentasi data atau pra-pemrosesan gambar untuk membuat sistem lebih tahan terhadap berbagai kondisi pencahayaan dan oklusi wajah sebagian.
4. **Uji Coba Pengguna (User Study):** Melakukan studi formal dengan beberapa pengguna untuk mengumpulkan umpan balik kuantitatif dan kualitatif mengenai kenyamanan dan pengalaman pengguna.

**DAFTAR PUSTAKA**


- Google. (s.d.). *MediaPipe Face Mesh*. Diakses dari https://ai.google.dev/edge/mediapipe/solutions/face_mesh
- Pedregosa, F., Varoquaux, G., Gramfort, A., dkk. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research.
- (Tambahkan referensi lain yang mungkin Anda gunakan)

**LAMPIRAN**

- **Repositori Kode Proyek:** [Link ke Repositori GitHub Anda]
- **Video Demonstrasi Proyek:** [Link ke Video YouTube Anda]