PENERAPAN 3D *CONVOLUTIONAL NEURAL NETWORK* UNTUK DETEKSI KANTUK PADA PENGEMUDI MOBIL

TUGAS AKHIR

Nathanael Timotius Sutedja
1122052



INSTITUT
TEKNOLOGI
HARAPAN
BANGSA
Veritas vos liberabit

PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
TAHUN

PENERAPAN 3D *CONVOLUTIONAL NEURAL NETWORK* UNTUK DETEKSI KANTUK PADA PENGEMUDI MOBIL

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh
gelar sarjana dalam bidang Informatika

Nathanael Timotius Sutedja
1122052



INSTITUT
TEKNOLOGI
HARAPAN
BANGSA
Veritas vos liberabit

PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
TAHUN

ABSTRAK

Nama	:	Nathanael Timotius Sutedja
Program Studi	:	Informatika
Judul	:	Penerapan 3D *Convolutional Neural Network* untuk deteksi kantuk pada pengemudi mobil

Kantuk pada pengemudi merupakan salah satu faktor utama penyebab kecelakaan lalu lintas karena dapat menurunkan tingkat kewaspadaan, waktu reaksi, dan konsentrasi saat berkendara. Oleh karena itu, diperlukan sistem deteksi kantuk yang akurat, cepat, dan bersifat non-invasif. Penelitian ini mengusulkan penerapan metode 3D *Convolutional Neural Network* (3D CNN) untuk mendeteksi kondisi kantuk pengemudi berdasarkan informasi visual dari video wajah pengemudi. Pada tahap awal, area wajah pengemudi diekstraksi menggunakan metode *Haar Cascade* untuk memperoleh *region of interest* sebelum diproses lebih lanjut oleh model 3D CNN. Model 3D CNN digunakan untuk menangkap informasi spasial dan temporal secara bersamaan dari rangkaian frame video. Penelitian ini juga menganalisis pengaruh pengaturan parameter *depth* dan *strides* terhadap kinerja model. Pengujian dilakukan menggunakan *UTA Real-Life Drowsiness Dataset* dengan skema validasi berbasis beberapa fold. Hasil penelitian menunjukkan bahwa penggunaan 3D CNN mampu mendeteksi kantuk pengemudi dengan tingkat akurasi yang baik. Selain itu, variasi nilai *depth* dan *strides* terbukti memengaruhi performa model secara signifikan. Model dengan konfigurasi parameter tertentu menghasilkan akurasi yang lebih optimal dibandingkan konfigurasi lainnya. Dengan demikian, sistem yang diusulkan berpotensi digunakan sebagai solusi deteksi kantuk pengemudi secara *real-time* untuk meningkatkan keselamatan berkendara.

Kata kunci: Deteksi Kantuk, 3D *Convolutional Neural Network*, *Computer Vision*, *Haar Cascade*, Pengemudi Mobil

ABSTRACT

Name	:	Nathanael Timotius Sutedja
Department	:	*Informatics*
Title	:	*Implementation of 3D Convolutional Neural Network for Detecting Driver Drowsiness*

Driver drowsiness is one of the major causes of traffic accidents as it reduces alertness, reaction time, and concentration while driving. Therefore, an accurate, fast, and non-invasive drowsiness detection system is required. This study proposes the implementation of a 3D Convolutional Neural Network (3D CNN) to detect driver drowsiness based on visual information extracted from facial video data. In the initial stage, the driver's facial region is extracted using the Haar Cascade method to obtain the region of interest before further processing. The 3D CNN model is employed to capture both spatial and temporal features from a sequence of video frames simultaneously. This research also analyzes the effect of depth and strides parameters on the model's performance. Experiments are conducted using the UTA Real-Life Drowsiness Dataset with a multi-fold validation scheme. The experimental results indicate that the proposed 3D CNN model is capable of detecting driver drowsiness with satisfactory accuracy. Furthermore, variations in depth and strides significantly affect the performance of the model. Certain parameter configurations achieve better accuracy compared to others. Therefore, the proposed system shows potential for real-time driver drowsiness detection applications to enhance road safety.

Drowsiness Detection, 3D Convolutional Neural Network, Computer Vision, Haar Cascade, Driver Monitoring.

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, penulis dapat menyelesaikan Tugas Akhir yang berjudul "Penerapan 3D Convolutional Neural Network untuk Deteksi Kantuk pada Pengemudi Mobil" dengan baik. Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Informatika pada Program Studi Informatika, Institut Teknologi Harapan Bangsa. Penyusunan Tugas Akhir ini bertujuan untuk mengkaji dan mengimplementasikan metode deep learning, khususnya 3D Convolutional Neural Network, dalam mendeteksi kondisi kantuk pengemudi secara non-invasif melalui pendekatan computer vision. Penulis berharap penelitian ini dapat memberikan kontribusi dalam pengembangan sistem keselamatan berkendara serta menjadi referensi bagi penelitian selanjutnya di bidang deteksi kantuk dan pengolahan citra berbasis video. Dalam proses penyusunan Tugas Akhir ini, penulis menyadari bahwa penyelesaian karya ini tidak terlepas dari bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada dosen pembimbing yang telah memberikan arahan dan masukan selama proses penelitian, serta kepada seluruh dosen Program Studi Informatika Institut Teknologi Harapan Bangsa atas ilmu yang telah diberikan. Penulis juga mengucapkan terima kasih kepada keluarga dan teman-teman yang telah memberikan dukungan moral selama proses penyusunan Tugas Akhir ini. Penulis menyadari bahwa Tugas Akhir ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun demi penyempurnaan penelitian ini di masa mendatang. Semoga Tugas Akhir ini dapat memberikan manfaat bagi pembaca dan perkembangan ilmu pengetahuan di bidang teknologi informasi.

Bandung, Tanggal Bulan Tahun

Hormat penulis,



Nathanael Timotius Sutedja

DAFTAR ISI

ABSTRAK **iv**

ABSTRACT **v**

KATA PENGANTAR **vi**

DAFTAR ISI **vii**

DAFTAR TABEL **x**

DAFTAR GAMBAR **xi**

DAFTAR ALGORITMA **xi**

BAB 1 PENDAHULUAN **1-1**

1.1 Latar Belakang 1-1
1.2 Rumusan Masalah 1-4
1.3 Tujuan Penelitian 1-4
1.4 Batasan Masalah 1-4
1.5 Konstribusi Penelitian 1-5
1.6 Metodologi Penelitian 1-5
1.7 Sistematika Pembahasan 1-5

BAB 2 LANDASAN TEORI **2-1**

2.1 Tinjauan Pustaka 2-1
2.1.1 Citra Digital 2-1
2.1.2 Pengolahan Citra 2-2
2.1.3 Haar Cascade 2-2
2.1.4 Deep Learning 2-3
2.1.5 *Convolutional Neural Network* 2-4
2.1.6 *3D Convolutional Neural Network* 2-5
2.1.6.1 *3D Convolutional Layer* 2-6
2.1.6.2 *Rectified Linear Unit Layer* (ReLU) 2-7
2.1.6.3 *3D Pooling Layer* 2-8
2.1.6.4 *Dropout Layer* 2-10
2.1.6.5 *Fully Connected Layer* 2-11

2.1.6.6 *Softmax Layer* 2-12
2.1.7 *Confusion Matrix* 2-13
2.1.8 Pustaka Pendukung 2-14
2.1.8.1 TensorFlow 2-14
2.1.8.2 Keras 2-15
2.1.8.3 OpenCV 2-16
2.1.8.4 Scikit-Learn 2-17
2.2 Tinjauan Studi 2-18
2.3 Tinjauan Objek 2-22
2.3.1 Orang Kantuk 2-22
2.3.2 *Dataset* Deteksi Kantuk 2-23

BAB 3 ANALISIS DAN PERANCANGAN SISTEM 3-1
3.1 Analisis Masalah 3-1
3.2 Kerangka Pemikiran 3-2
3.3 Urutan Proses Global 3-4
3.4 Analisis Manual 3-5
3.4.1 *Data Sampling* 3-5
3.4.2 *3D Convolutional Neural Network* 3-9
3.4.3 Pengujian *Convolutional Neural Network* 3-21

BAB 4 IMPLEMENTASI DAN PENGUJIAN 4-1
4.1 Lingkungan Implementasi 4-1
4.1.1 Spesifikasi Perangkat Keras 4-1
4.1.2 Spesifikasi Perangkat Lunak 4-1
4.2 Implementasi Perangkat Lunak 4-1
4.2.1 Implementasi *Class* 4-1
4.2.1.1 *Class Preprocessing* 4-2
4.2.1.2 *Class 3D CNN Model* 4-2
4.2.1.3 *Class Evaluation* 4-2
4.2.2 Penggunaan Google Colaboratory 4-3
4.2.3 Penggunaan Dataset 4-3
4.3 Pengujian 4-3
4.4 Hasil Pengujian 4-4
4.5 Pengujian dengan *dataset* sendiri 4-8
4.6 Pembahasan Hasil Pengujian 4-9
4.6.1 Analisis Umum 4-10
4.6.2 Pembahasan Pengaruh Depth pada Model 3D CNN 4-10

4.6.3 Pembahasan Pengaruh Strides pada Model 3D CNN 4-10
4.6.4 Pembahasan Pengaruh Pengunaan Haar Cascade 4-10
4.6.5 Perbandingan Keseluruhan 4-11
4.7 Analisis Kesalahan 4-12

BAB 5 KESIMPULAN DAN SARAN **5-1**
5.1 Kesimpulan 5-1
5.2 Saran 5-2

DAFTAR TABEL

2.1 Daftar *method* yang digunakan dari pustaka TensorFlow 2-14
2.2 Daftar *method* yang digunakan dari pustaka Keras 2-15
2.3 Daftar *method* yang digunakan dari pustaka OpenCV 2-16
2.4 Daftar *method* yang digunakan dari pustaka Scikit-Learn 2-17
2.5 Tinjauan Studi 2-18

3.1 Arsitektur *3D-CNN* Untuk Depth=64 3-10
3.2 **3.2** Perhitungan *Standard Normal Distribution* 3-14
3.3 **3.3** Perhitungan *feature map* 3D 3-15
3.4 **3.4** Perhitungan *max pooling* 3D CNN 3-18
3.5 **3.5** Perhitungan *mean* 3-18
3.6 **3.6** Perhitungan standar deviasi 3-18
3.7 **3.7** Perhitungan variance 3-19
3.8 **3.8** Perhitungan *Global Average Pooling* 3-19
3.9 **3.9** Perhitungan *Dropout* 3-20
3.10 **3.10** Perhitungan *fully connected* 3-20
3.11 **3.11** Perhitungan *softmax* 3-20

4.1 Daftar method pada *Class* Video Preprocessing 4-2
4.2 Daftar method pada class 3D CNN Model 4-2
4.3 Daftar method pada class Evaluation 4-2
4.4 Parameter Pengujian Model 4-4
4.5 Hasil Pengujian Model Berdasarkan Strides dan Depth untuk Setiap Fold 4-4
4.6 Hasil Pengujian Model Yang Tidak Menggunakan Haar 4-6
4.7 Rata-Rata Hasil Pengujian Model dari setiap *fold* berdasarkan Parameter *Strides* dan *Depth* 4-7
4.8 Rata-Rata Hasil Pengujian Model Yang Tidak Menggunakan Haar . 4-7
4.9 Confusion Matrix Klasifikasi Drowsiness 4-9

DAFTAR GAMBAR

2.1 Representasi Citra Digital sebagai Array Dua Dimensi [9] 2-1
2.2 Arsitektur CNN [18] 2-4
2.3 Di dalam *convolutional network* [12] 2-5
2.4 Arsitektur 3D CNN Dasar[15] 2-6
2.5 *Max Pooling* [16] 2-9
2.6 Illustrasi *Dropout Layer* [19] 2-10
2.7 *Confusion Matrix* [20] 2-13
2.8 Contoh Dataset [33] 2-24
3.1 Kerangka Pemikiran 3-2
3.2 Urutan Proses Global 3-4
3.3 Contoh *frame drowsy* 3-6
3.4 Contoh *frame not drowsy* 3-6
3.5 Hasil *haar cascade* 3-7
3.6 Contoh *resize* citra 3-8
3.7 Contoh Array video 3-12
3.8 Contoh Array video 3-13
3.9 Contoh Padding Pada Array video 3-13
3.10 Contoh Array Kernel 3x3x3x2 3-14
3.11 Contoh perhitungan *feature map* pada indeks 0,0,0 3-16
3.12 Contoh perhitungan *feature map* pada indeks 0,0,1 3-16
3.13 Hasil *feature map* 3-17
3.14 Contoh hasil ReLu 3-17
3.15 Ilustrasi pengerjaan *max pooling* 3-17
3.16 Hasil Max Pooling 3-18
3.17 Hasil Normalisasi 3-19
3.18 Contoh *Confusion Matrix* 3-21
4.1 Pengaruh depth dan strides yang mengggunakan haar 4-8
4.2 Dataset sendiri kantuk 4-8
4.3 Dataset sendiri tidak kantuk 4-9
4.4 Hubungan Depth Terhadap Akurasi untuk Setiap Stride 4-12
4.5 Hasil deteksi haar *frontal faces* yang gagal 4-13

DAFTAR ALGORITMA

3.1 Extract faces untuk Depth=64 3-8
3.2 Training dan testing model 3-9
3.3 Build_3D_CNN untuk Strides=2 3-10

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kantuk, yang diartikan sebagai keinginan untuk tidur, dapat menimbulkan berbagai gejala seperti penurunan waktu reaksi, gangguan kesadaran, atau terjadinya *microsleeps* (tertidur sejenak selama lebih dari 500 milidetik). Kekurangan tidur menjadi masalah besar bagi banyak pengemudi di jalan raya, seperti sopir truk, sopir *travel*, dan orang yang melakukan perjalanan jauh, yang semuanya berisiko tinggi. Selain itu, rasa kantuk mengurangi konsentrasi pengemudi, yang mengarah pada kondisi berbahaya. Hal ini secara signifikan meningkatkan kemungkinan pengemudi melewatkan rambu-rambu jalan, keluar jalur, atau berpindah ke jalur lain, bahkan dapat menyebabkan kecelakaan [1]. Kantuk saat berkendara merupakan salah satu faktor utama penyebab kecelakaan lalu lintas di seluruh dunia. Menurut laporan *National Sleep Foundation*, sekitar 20% pengemudi pernah mengalami kantuk selama mengemudi, yang menyebabkan ratusan ribu kecelakaan setiap tahunnya [2]. Untuk menangani masalah ini, dibutuhkan sistem deteksi kantuk yang mampu memberikan peringatan dini dengan cepat dan tepat untuk meningkatkan keselamatan pengemudi dan pengguna jalan lainnya.

Berbagai metode telah dikembangkan untuk mendeteksi kantuk, yang dapat dikelompokkan menjadi tiga kategori utama. Pertama, metode berbasis kendaraan yang memonitor perilaku seperti pergerakan setir, penyimpangan jalur, atau pola kecepatan kendaraan. Kedua, metode berbasis sensor fisiologis yang mengukur sinyal tubuh seperti *elektrooculogram* (EOG), *electroencephalogram* (EEG), atau detak jantung untuk mendeteksi tingkat kewaspadaan pengemudi. Ketiga, metode berbasis *computer vision* yang menggunakan kamera untuk menganalisis pola visual seperti kedipan mata, ekspresi wajah, dan gerakan kepala [3].

Meskipun masing-masing metode memiliki keunggulan, ada juga keterbatasan yang signifikan. Metode berbasis kendaraan cenderung murah dan mudah diimplementasikan, tetapi hanya mampu menganalisis perilaku kendaraan, bukan kondisi pengemudi secara langsung. Hal ini dapat menghasilkan deteksi yang kurang akurat dalam situasi tertentu. Metode berbasis fisiologis, seperti penggunaan EEG dan EOG, menawarkan akurasi tinggi tetapi mahal, invasif, dan memerlukan perangkat keras tambahan yang dapat mengganggu kenyamanan

pengemudi. Sebaliknya, metode berbasis *computer vision* bersifat non-invasif, dapat bekerja secara *real-time*, dan murah. Namun, metode ini masih menghadapi tantangan seperti sudut pandang kamera, dan keberadaan aksesori seperti kacamata [3]. Mitigasi yang dilakukan untuk mengatasi sudut pandang adalah menggunakkan 3D CNN untuk menangkap informasi spasial-temporal sedangkan untuk aksesori seperti kacamata digunakan mulut sebagai indikasi untuk mendeteksi.

Dalam pendekatan *computer vision*, berbagai algoritma *machine learning* dan *deep learning* telah digunakan untuk mendeteksi kantuk pengemudi. *Convolutional Neural Network* (CNN) adalah salah satu metode *deep learning* yang sangat efektif untuk ekstraksi fitur dari citra, seperti mendeteksi kondisi mata (terbuka atau tertutup) dan aktivitas mulut (menguap atau tidak). CNN mampu mempelajari pola visual kompleks dari data citra dengan akurasi 97%. Model CNN yang lebih kompleks seperti VGG16 juga sering digunakan untuk meningkatkan akurasi deteksi melalui arsitektur jaringan yang lebih dalam dan mendetail. Namun, VGG16 memiliki akurasi 74% [1].

Penelitian oleh Chen et al [4] mengusulkan pendekatan *hybrid* yang menggabungkan CNN dan *Long Short-Term Memory* (LSTM) untuk meningkatkan akurasi deteksi. CNN digunakan untuk mengekstrak fitur mendalam dari area mata dan mulut dengan akurasi mencapai 93.3% untuk deteksi kondisi mulut dan 84.9% untuk deteksi kondisi mata. Model SqueezeNet yang dimodifikasi menunjukkan performa optimal untuk deteksi mulut, sementara arsitektur berbasis VGG-16 digunakan untuk deteksi kondisi mata. Untuk menangkap informasi temporal, output dari CNN kemudian diproses oleh LSTM yang mampu memodelkan perubahan keadaan kantuk secara berurutan. Pendekatan ini dilengkapi dengan metode *Factorized Bilinear Feature Fusion* (FBFF) yang memungkinkan fusi non-linear antara fitur mata dan mulut, menghasilkan akurasi deteksi kantuk sebesar 73.8%.

Penelitian oleh Wijnands et al, mengusulkan metode deteksi kantuk berbasis *depthwise separable 3D Convolutional Neural Network* (3D-CNN) yang mampu memanfaatkan informasi spasial dan temporal secara bersamaan. Model ini mencapai akurasi 73.9% pada *Driver Drowsiness Detection Dataset* (DDD) dan dapat berjalan secara *real-time* di perangkat ponsel, sehingga efektif digunakan dalam kondisi nyata [5].

Selain itu, penelitian oleh Xiang et al. mengusulkan sistem deteksi kantuk

berbasis kombinasi *multi-branch* 3D CNN dan *attention mechanism*. Model ini mengekstraksi informasi spasial dan temporal dari wajah pengemudi, kemudian mengoptimalkan bobot fitur dengan *Squeeze-and-Excitation* (SE) *module*. Indikator kantuk diukur menggunakan *Eye Aspect Ratio* (EAR) dan *Mouth Aspect Ratio* (MAR), lalu diklasifikasikan dengan *Support Vector Machine* (SVM). Hasil eksperimen menunjukkan akurasi mencapai lebih dari 95% pada beberapa *dataset* standar, bahkan lebih unggul dibandingkan model CNN klasik seperti ResNet-50, MobileNetV3, dan EfficientNet [6].

Penelitian oleh Sedik et al. mengusulkan sistem deteksi kantuk adaptif berbasis 3D CNN dan *ensemble models* dengan augmentasi data menggunakan *Conditional* GAN. Model ini diuji pada *dataset* DROZY dan *dataset* citra gejala kantuk, menghasilkan akurasi hingga 99% pada *dataset* video dengan 3D CNN, dan 98% pada *dataset* citra, serta 98% pada kombinasi keduanya, melampaui akurasi penelitian sebelumnya yang hanya mencapai 97%. Hasil ini menunjukkan efektivitas 3D CNN dalam menangkap fitur spasial-temporal secara *real-time* untuk deteksi kantuk pengemudi [7].

Salah satu langkah penting dalam meningkatkan performa model *deep learning* adalah augmentasi data. Augmentasi data digunakan untuk meningkatkan variasi citra dalam *dataset*, sehingga model lebih tahan terhadap kondisi yang beragam. Sebagai contoh, penelitian yang dilakukan oleh Magán et al. (2022) menggunakan augmentasi seperti rotasi, dan *horizontal flipping*. Teknik ini membantu model mempelajari pola yang lebih luas dan mengurangi risiko *overfitting* ketika jumlah *dataset* terbatas. Selain itu, augmentasi data juga memungkinkan model untuk bekerja lebih baik dalam situasi dunia nyata, seperti ketika sudut pandang kamera berubah atau ketika terdapat gangguan pada citra seperti *noise* [8].

Namun, meskipun telah banyak metode yang digunakan, masih ada beberapa tantangan besar. Salah satu masalah utama adalah metode berbasis *computer vision* sering mengalami alarm palsu, misalnya ketika pengemudi memejamkan mata sejenak yang tidak terkait dengan kantuk. Hal ini menggarisbawahi kebutuhan akan sistem yang tidak hanya akurat tetapi juga tahan terhadap variasi kondisi dunia nyata. Untuk mengatasi tantangan ini, penelitian ini mengusulkan penggunaan 3D CNN karena mampu memperhitungkan informasi spasial dan temporal dalam video atau urutan citra, sehingga bisa membedakan antara aksi seperti memejamkan mata yang sesaat dan kondisi yang lebih berbahaya seperti kantuk.

Penelitian ini bertujuan untuk mengembangkan sistem deteksi kantuk dengan menggunakan 3D CNN yang bekerja secara *real-time*, akurat, dan non-invasif. Sistem ini dirancang untuk mengidentifikasi tanda-tanda kantuk berdasarkan kondisi mata pengemudi dan sistem akan diuji pada *dataset* publik. Penelitian ini juga akan mengevaluasi pengaruh *depth* dan *stride* terhadap kinerja model. Dengan pendekatan ini, penelitian ini diharapkan dapat menghasilkan solusi yang baik dan dapat memberikan kontribusi penting dalam meningkatkan keselamatan jalan raya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah disebutkan, terdapat rumusan masalah sebagai berikut.

1. Berapa tingkat akurasi yang dihasilkan oleh model dalam memprediksi kondisi kantuk pengemudi menggunakan metode 3D CNN?
2. Bagaimana pengaruh pengaturan *indicator* seperti *depth*, dan *strides* terhadap tingkat akurasi sistem deteksi kantuk pengemudi menggunakan metode 3D CNN?
3. Bagaimana pengaruh *haar_cascades* terhadap tingkat akurasi dan waktu pada sistem deteksi kantuk menggunakan metode 3D CNN?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengukur dan mengevaluasi tingkat akurasi yang dihasilkan oleh model dalam memprediksi kondisi kantuk pengemudi menggunakan metode 3D CNN.
2. Menganalisis pengaruh nilai *indicator* seperti jumlah *depth*, *strides*, dan *haar_cascades*, terhadap tingkat akurasi sistem deteksi kantuk pengemudi menggunakan metode 3D CNN.
3. Menganalisis pengaruh *haar_cascades* terhadap akurasi dan waktu untuk deteksi kantuk pengemudi menggunakan metode 3D CNN.

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem hanya menggunakan metode berbasis visi komputer untuk mendeteksi kantuk tanpa melibatkan data dari sensor fisiologis atau perilaku kendaraan.
2. Penelitian ini terbatas pada penggunaan dataset yang tersedia, yang mungkin tidak sepenuhnya mewakili semua kondisi dan variasi perilaku pengemudi

dalam situasi dunia nyata. Keterbatasan ini dapat mempengaruhi akurasi dan generalisasi sistem deteksi kantuk.

1.5 Konstribusi Penelitian

Penelitian ini memberikan kontribusi sebagai berikut:

1. Mengembangkan sistem deteksi kantuk berbasis 3D CNN yang bekerja secara *real-time*, akurat, dan non-invasif.
2. Mengoptimalkan nilai *indicator* seperti *depth*, dan *strides* untuk meningkatkan akurasi sistem deteksi kantuk.

1.6 Metodologi Penelitian

Metodologi penelitian yang dilakukan dalam penelitian ini adalah sebagai berikut.

1. Studi Literatur.
   Penelitian ini dimulai dengan studi literatur yaitu mengumpulkan bahan-bahan referensi dari berbagai sumber yang membahas metode-metode deteksi kantuk.
2. Pengambilan Sampel.
   Data sampel yang digunakan diambil dari kaggle yang berjudul "*UTA Real-Life Drowsiness Dataset*" dari kaggle.
3. Analisis Masalah.
   Pada tahap ini dilakukan analisis permasalahan, termasuk identifikasi batasan-batasan yang ada, serta kebutuhan yang diperlukan untuk menyelesaikan permasalahan yang telah diartikan.
4. Perancangan.
   Pada tahap ini dilakukan penyusunan rancangan sistem yang mencakup algoritma 3D CNN untuk mencari solusi yang tepat dalam menyelesaikan masalah yang telah diidentifikasi.
5. Implementasi.
   Pada tahap ini dilakukan implementasi hasil analisis dari perancangan ke dalam sistem deteksi kantuk yang telah dirancang dan pengujian terhadap *dataset*.
6. Evaluasi Metode (Kesimpulan).
   Pada tahap ini dilakukan penilaian terhadap kinerja deteksi kantuk yang telah diimplementasikan berdasarkan data penelitian yang terkumpul.

1.7 Sistematika Pembahasan

Bab 1: Pendahuluan. Bab ini menjelaskan latar belakang, rumusan masalah, tujuan penelitian, kontribusi penelitian, dan metode penelitian yang difokuskan pada penggunaan 3D CNN dalam deteksi kantuk.

Bab 2: Landasan Teori. Bab ini menjelaskan teori mengenai metode 3D CNN yang digunakan.

Bab 3: Analisis dan Perancangan. Bab ini menjelaskan analisis terhadap permasalahan kantuk pada pengemudi mobil, solusi yang diusulkan dengan pendekatan 3D CNN.

Bab 4: Implementasi dan Pengujian. Bab ini menjelaskan proses implementasi dan pengujian terhadap aplikasi yang dibuat beserta evaluasinya.

Bab 5: Kesimpulan dan Saran. Bab ini menjelaskan kesimpulan dari aplikasi yang telah dibuat dan saran untuk pengembangan aplikasi serupa di masa mendatang.

BAB 2 LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada bagian ini, akan dibahas berbagai penelitian dan teori yang relevan dengan topik penelitian ini. Tinjauan pustaka bertujuan untuk memberikan dasar teori yang mendukung pemahaman terhadap permasalahan yang diteliti. Teori-teori yang digunakan bersumber dari buku referensi serta jurnal penelitian terkait.

2.1.1 Citra Digital

Citra digital adalah representasi dua dimensi dari suatu fungsi $f(x,y)$, di mana x dan y menunjukkan koordinat spasial, sedangkan f menggambarkan intensitas atau tingkat keabuan pada titik koordinat tersebut. Sebuah citra dikatakan digital jika nilai x, y, dan f bersifat diskret dan terbatas. Piksel merupakan elemen terkecil dari citra digital dan menjadi dasar untuk pengolahan citra. Proses untuk mengubah citra menjadi bentuk digital melibatkan dua tahap utama, yaitu sampling spasial untuk menentukan resolusi spasial citra, serta kuantisasi untuk menetapkan resolusi intensitas. Citra digital memiliki berbagai aplikasi, salah satunya dalam teknologi pemantauan visual, seperti deteksi kantuk. Dalam aplikasi ini, citra digital digunakan untuk mengidentifikasi perubahan tertentu, seperti ekspresi wajah yang menandakan kelelahan atau pergerakan mata yang tidak biasa [9].

Citra digital dapat digambarkan dalam bentuk array dua dimensi, seperti yang ditunjukkan pada Gambar 2.1. Representasi ini memungkinkan citra diproses secara matematis sebagai matriks, di mana setiap elemen matriks $f(i,j)$ mencerminkan tingkat intensitas piksel pada posisi (i,j). Intensitas ini biasanya direpresentasikan dalam nilai diskret, seperti tingkat keabuan atau nilai warna tertentu [9].

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.1 Representasi Citra Digital sebagai Array Dua Dimensi [9]

Di mana	:
$f(x,y)$	: Nilai dari sebuah piksel
x	: Posisi bari
y	: Posisi kolom
M	: Jumlah kolom dalam array citra.
N	: Jumlah baris dalam array citra

Gambar 2.1 memberikan contoh citra sederhana dalam format matriks kecil, yang digunakan untuk memudahkan pemahaman konsep ini. Setiap elemen matriks menunjukkan tingkat kecerahan pada lokasi piksel tertentu dalam citra. Matriks dengan ukuran yang lebih besar, yang memiliki lebih banyak elemen, memberikan resolusi yang lebih tinggi, sehingga detail citra menjadi lebih tajam dan jelas. Namun, peningkatan resolusi ini memerlukan lebih banyak kapasitas penyimpanan dan sumber daya komputasi yang lebih besar untuk pemrosesan [9].

2.1.2 Pengolahan Citra

Pengolahan citra adalah proses manipulasi citra digital untuk meningkatkan kualitas visual, mengekstrak informasi tertentu, atau mempersiapkan citra untuk analisis lebih lanjut. Pengolahan citra mencakup berbagai teknik yang bertujuan untuk memanipulasi representasi numerik dari citra dalam bentuk matriks piksel. Proses ini mencakup tahapan seperti peningkatan kualitas citra (*image enhancement*), restorasi citra (*image restoration*), analisis informasi dalam citra, dan transformasi geometris [9].

2.1.3 Haar Cascade

Haar Cascade Classifier merupakan metode deteksi objek yang efisien dan cepat, khususnya untuk mendeteksi wajah dan mata secara *real-time* dalam pengolahan citra. Metode ini dikembangkan oleh Paul Viola dan Michael Jones pada tahun 2001 melalui algoritma Viola-Jones, yang memanfaatkan fitur sederhana berbasis pembelajaran mesin untuk mengklasifikasikan objek dari gambar. Metode ini bekerja berdasarkan ekstraksi fitur *Haar-like*. Setiap fitur *Haar-like* dihitung sebagai selisih intensitas piksel antara area putih dan hitam dalam persegi panjang. Metode ini bekerja berdasarkan ekstraksi fitur *Haar-like*. Setiap fitur *Haar-like* dihitung sebagai selisih intensitas piksel antara area putih dan hitam dalam persegi panjang [10].

Untuk efisiensi lebih lanjut, *classifier* disusun dalam bentuk *cascade* tahap-tahap. Setiap tahap berisi beberapa fitur. Jika jendela citra gagal pada tahap awal,

jendela tersebut langsung dibuang tanpa memproses tahap selanjutnya. Ini membuat deteksi sangat cepat, ideal untuk aplikasi *real-time* [10].

Haar Cascade merupakan salah satu metode klasik dalam pendeteksian objek yang dikenal karena kecepatan pemrosesannya. Keunggulan utama *Haar Cascade* terletak pada kesederhanaan fitur *Haar* (*edge*, *line*, dan *four-rectangle features*) serta penggunaan *cascading window*. Setiap jendela pemindaian dievaluasi secara bertahap, di mana tahap awal hanya menggunakan fitur sederhana untuk menyingkirkan area yang tidak relevan. Dengan demikian, hanya sebagian kecil area citra yang diproses lebih lanjut, sehingga memungkinkan pendeteksian secara *real-time*. Selain itu, penerapan *integral image* membuat perhitungan fitur menjadi sangat efisien dan lebih cepat [11].

Namun, *Haar Cascade* memiliki beberapa kelemahan signifikan. Metode ini cenderung menghasilkan *false positive* yang tinggi, terutama pada kondisi pencahayaan yang bervariasi atau latar belakang yang kompleks. Meskipun fitur-fiturnya sederhana, proses pelatihan dan pemindaian multi-skala dapat menjadi komputasional intensif. Selain itu, akurasinya relatif lebih rendah dibandingkan metode deteksi modern berbasis *deep learning*. Selain itu pada penggunaan *frontal face classifier* pada metode *Haar Cascade* memiliki keterbatasan yang signifikan. Hal ini hanya dilatih untuk mendeteksi wajah dalam posisi menghadap ke depan, sehingga tidak mampu mendeteksi wajah dengan variasi pose, seperti wajah yang menghadap ke samping, menunduk, atau miring. Akibatnya, sistem pendeteksian menjadi kurang *robust* terhadap perubahan orientasi wajah, yang dapat menyebabkan kegagalan deteksi (*false negative*) pada kondisi nyata yang dinamis [11].

2.1.4 Deep Learning

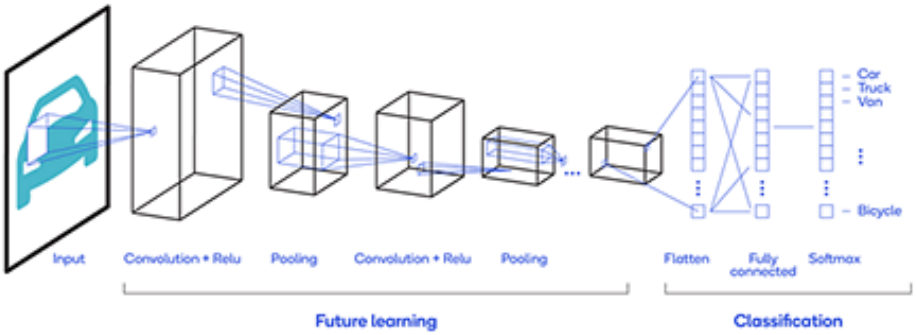
Deep learning adalah cabang dari pembelajaran *machine learning* yang memungkinkan sistem komputasi untuk belajar representasi data melalui berbagai lapisan abstraksi. Metode ini terdiri dari jaringan modular yang memanfaatkan banyak lapisan pemrosesan non-linear untuk mentransformasikan representasi data awal ke representasi tingkat tinggi. Hal ini memungkinkan sistem untuk mengenali pola-pola kompleks dalam data berdimensi tinggi, seperti citra, video, atau sinyal suara. *Deep learning* mempunyai ciri utama yaitu kemampuannya untuk secara otomatis belajar fitur dari data mentah tanpa memerlukan rekayasa fitur manual yang memakan waktu dan membutuhkan keahlian domain [12].

Deep learning memiliki keunggulan signifikan dibandingkan dengan

metode pembelajaran mesin tradisional. Salah satunya adalah kemampuannya untuk secara otomatis mengekstraksi fitur penting dari data mentah melalui pendekatan hierarkis, tanpa memerlukan proses manual seperti pada pembelajaran mesin konvensional. Selain itu, *deep learning* dapat langsung memproses data mentah, seperti nilai piksel dalam citra, dan belajar merepresentasikan data tersebut melalui lapisan-lapisan non-linear, mengurangi kebutuhan pra-pemrosesan. Teknologi ini juga lebih efektif dalam menangani data berdimensi tinggi, seperti citra, video dan suara, yang sering sulit diolah oleh metode tradisional [12].

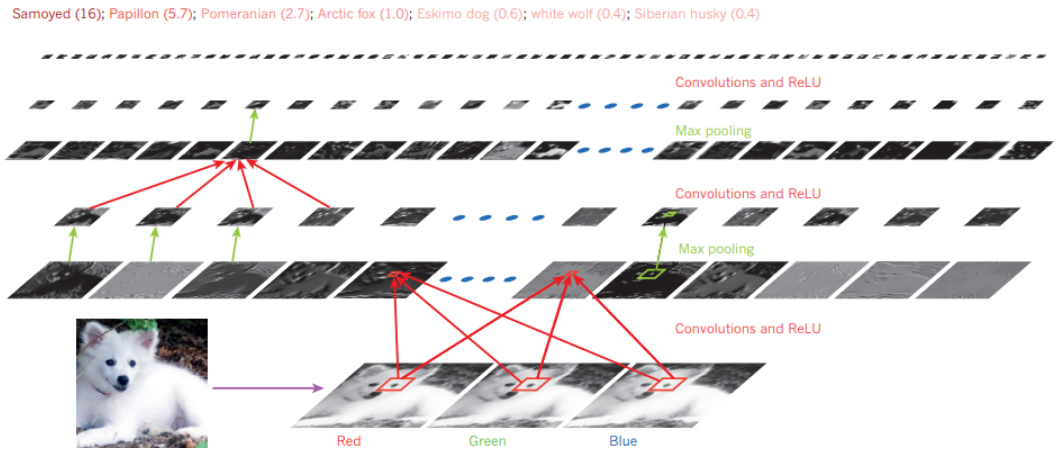
2.1.5 *Convolutional Neural Network*

Convolutional Neural Networks (CNN) dirancang khusus untuk mengolah data yang berbentuk array, seperti citra yang terdiri dari beberapa kanal warna (misalnya RGB). CNN memanfaatkan beberapa konsep utama, seperti *local connectivity*, *shared weight*, dan *pooling layer*, untuk mengidentifikasi pola-pola lokal dalam data. Dalam hal ini, unit pada *convolution layer* hanya terhubung dengan subset lokal piksel dari lapisan sebelumnya, yang membantu mengurangi jumlah parameter yang perlu dilatih. Selain itu, CNN menggunakan prinsip *shared weight*, di mana setiap unit dalam *feature maps* berbagi filter yang sama, memungkinkan deteksi pola yang serupa di berbagai posisi dalam citra. Untuk mengurangi dimensi data dan meningkatkan ketahanan terhadap pergeseran kecil, *pooling layer*, seperti *max pooling*, digunakan untuk memilih nilai maksimum dari suatu wilayah. Keunggulan utama CNN adalah kemampuannya untuk secara otomatis belajar dan mengidentifikasi fitur-fitur penting, menjadikannya sangat efektif dalam tugas pengenalan objek pada citra dan video. Berbagai aplikasi CNN termasuk pengenalan wajah, deteksi objek, serta pengelompokan wilayah biologis dalam citra medis [12] [13].



Gambar 2.2 Arsitektur CNN [18]

Gambar 2.2 menggambarkan arsitektur dasar *Convolutional Neural Network* (CNN), yang terdiri dari tiga komponen utama: lapisan konvolusi, *pooling*, dan *fully connected*. Lapisan konvolusi mendeteksi pola lokal seperti tepi melalui filter, sementara lapisan *pooling* mengurangi dimensi data dengan memilih nilai maksimum atau rata-rata untuk meningkatkan efisiensi dan ketahanan terhadap perubahan kecil. Pada tahap akhir, lapisan *fully connected* menggabungkan informasi dari lapisan sebelumnya untuk menghasilkan output berupa skor probabilitas untuk klasifikasi objek. Gambar 2.2 menunjukkan alur data dari citra input hingga output berupa prediksi kelas dengan probabilitas.



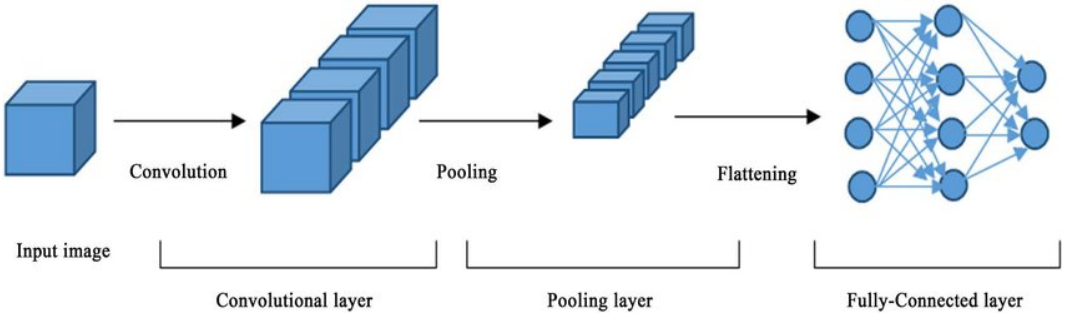
Gambar 2.3 Di dalam *convolutional network* [12]

Gambar 2.3 menunjukkan bagaimana CNN memproses data dari gambar seekor anjing hingga menghasilkan *output* berupa prediksi kelas objek, mengilustrasikan hierarki pemrosesan fitur dari tingkat rendah ke tinggi. Pendekatan ini membedakan CNN dari teknik pembelajaran mesin tradisional, yang sering kali membutuhkan rekayasa fitur manual. Dengan keunggulan otomatisasi dan efisiensi, CNN telah merevolusi berbagai bidang yang memanfaatkan data visual.

2.1.6 *3D Convolutional Neural Network*

3D *Convolutional Neural Network* (3D CNN) merupakan pengembangan dari arsitektur CNN 2D yang dirancang khusus untuk memproses data volumetrik atau tiga dimensi. Berbeda dengan CNN 2D yang hanya melakukan konvolusi pada dua dimensi yaitu *width*, dan *height*, 3D CNN melakukan operasi konvolusi

pada tiga dimensi, yaitu *width*, *height*, dan *depth*. Dengan demikian, 3D CNN mampu menangkap informasi spasial dalam ketiga dimensi secara bersamaan, yang menjadikannya sangat efektif dalam memproses data video. Gambar 2.4 menunjukkan proses dari sebuah 3D CNN secara umum [14].



Gambar 2.4 Arsitektur 3D CNN Dasar[15]

Secara struktural, 3D CNN memiliki lapisan-lapisan utama yang serupa dengan CNN konvensional, yaitu *Convolution Layer*, *pooling layer*, dan *fully-connected layer*. Namun, semua operasi dilakukan dalam ruang tiga dimensi. *Convolution layer* menggunakan kernel 3D untuk mengekstraksi fitur spasial dari volume data, sementara *pooling layer* mengurangi ukuran spasial dalam tiga arah untuk mengurangi kompleksitas komputasi. *Fully-connected layer* digunakan untuk menghasilkan prediksi akhir, seperti klasifikasi [14].

2.1.6.1 *3D Convolutional Layer*

3D Convolutional layer adalah elemen utama dalam 3D CNN yang dirancang untuk memproses data berdimensi tiga, seperti video. Lapisan ini bekerja dengan melakukan operasi matematis yang disebut konvolusi untuk mengekstraksi fitur-fitur lokal dari data masukan. Proses ini memungkinkan jaringan untuk mengenali pola-pola seperti tepi, tekstur, atau bentuk-bentuk tertentu dalam data. Berbeda dengan 2D *convolution* yang hanya bergerak pada dua dimensi *width* dan *height*, 3D *convolution* memproses data pada tiga dimensi: *height*, *width*, dan *depth*. *Depth* ini bisa merepresentasikan rangkaian frame dalam video atau irisan dalam volume data. [16].

Operasi konvolusi melibatkan penggunaan filter 3D (atau *kernel* 3D), yaitu sebuah matriks berukuran kecil yang memiliki dimensi *height*, *width*, dan *depth*. Filter ini bergerak melintasi seluruh data masukan 3D, melakukan operasi matematis berupa *dot product* antara elemen-elemen filter dengan bagian-bagian data masukan yang sesuai. Dengan cara ini, 3D *convolution* dapat mengekstraksi fitur-fitur lokal yang mempertimbangkan konteks spasial dan temporal sekaligus,

seperti pola pergerakan objek dalam video [19]. Pada penelitian ini digunakan metode *Standard Normal Distirbution* yang dapat dilihat pada persamaan 2.1. Untuk proses konvolusi menggunakan persamaan 2.2.

$$k = \frac{1}{\sigma \times \sqrt{2\pi}} \times e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

Di mana	:
k	: Nilai *kernel*
x	: Variabel *random* normal
μ	: Nilai rata-rata
σ	: *Standard deviation*
π	: Nilai pi (3.14159)
e	: Nilai *euler* (2.7182818)

$$O(i,j,k) = \sum_{d=0}^{D-1} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} I(d+i, h+j, w+k) \cdot K(d,h,w) \tag{2.2}$$

Di mana	:
$O(i,j,k)$	: nilai output pada posisi (i,j,k) setelah operasi konvolusi.
$I(d+i,h+j,w+k)$	: nilai *input* pada posisi $(d+i,h+j,w+k)$.
$K(d,h,w)$	: elemen dari kernel pada posisi (d,h,w).
D,H,W	: ukuran kernel dalam dimensi *depth*, *height*, dan *width*, masing-masing.
d, h, dan w	: indeks dalam kernel, yang bervariasi dari 0 hingga $D-1$, $H-1$, dan $W-1$.
i, j, dan k	: posisi dalam output yang dihitung oleh konvolusi.

2.1.6.2 *Rectified Linear Unit Layer* (ReLU)

ReLU adalah salah satu fungsi aktivasi yang paling banyak digunakan dalam *Artifical Neural Network* (ANN), termasuk pada *hidden layer* dalam jaringan *feedforward*. ReLU berfungsi sebagai fungsi aktivasi yang mendefinisikan outputnya sebagai nilai maksimum antara nol dan input, hal ini dapat dilihat pada persamaan 2.3 [16].

$$f(x) = max(0,x) \tag{2.3}$$

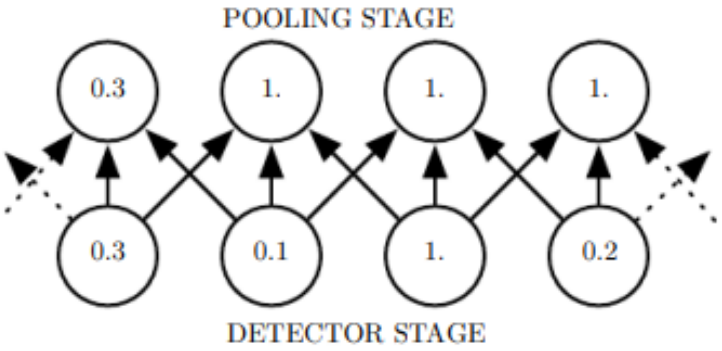
Di mana	:
$f(x)$	: Fungsi aktivasi ReLU
x	: Input ke neuron dalam jaringan saraf

ReLU memiliki beberapa keunggulan dibandingkan fungsi aktivasi lainnya, seperti sigmoid dan tanh, yang menjadikannya pilihan utama dalam jaringan saraf modern. Salah satu keunggulannya adalah kemudahan optimasi. Gradien dari ReLU tidak mengalami masalah saturasi seperti pada sigmoid, sehingga gradien tetap besar selama unit aktif, yaitu saat nilai input $z > 0$. Hal ini memungkinkan pembaruan bobot yang lebih efektif selama pelatihan. Selain itu, ReLU sangat efisien secara komputasi karena perhitungannya hanya melibatkan operasi sederhana, yaitu mengambil nilai maksimum antara nol dan input. Keunggulan lain adalah kemampuannya menghasilkan aktivasi sparse. Untuk sebagian besar input negatif, output ReLU adalah nol, yang membuat jaringan lebih hemat sumber daya dan efisien dalam menangkap pola penting tanpa menangani noise yang tidak relevan. Kombinasi dari sifat-sifat ini menjadikan ReLU fungsi aktivasi yang unggul untuk berbagai aplikasi *deep learning*.

2.1.6.3 *3D Pooling Layer*

3D Pooling Layer merupakan salah satu komponen penting dalam arsitektur 3D CNN. Lapisan ini berfungsi untuk mengurangi dimensi spasial dan temporal dari data masukan, dengan tetap mempertahankan fitur-fitur penting yang telah diekstraksi oleh lapisan *convolution* sebelumnya. Dengan kata lain, *pooling layer* membantu menyederhanakan representasi data tanpa kehilangan informasi yang relevan [16].

Pooling layer juga meningkatkan invariansi terhadap translasi, yang berarti model menjadi lebih *robust* terhadap pergeseran kecil pada data masukan. Hal ini sangat penting dalam pengolahan citra, di mana fitur-fitur seperti tepi atau pola sering kali tetap relevan meskipun terjadi pergeseran posisi objek [16]. Pada gambar 2.5 dijelaskan efek *max pooling* dalam meningkatkan invariansi translasi. Gambar 2.5 menggambarkan bagaimana *pooling layer* memproses data dengan mempertahankan nilai maksimum dari setiap jendela lokal dalam peta fitur, sehingga memungkinkan representasi data tetap stabil meskipun terjadi pergeseran kecil pada posisi fitur dalam input [16].



Gambar 2.5 *Max Pooling* [16]

Terdapat beberapa metode *pooling*, antara lain max *pooling*, *average pooling*. *Max pooling* memilih nilai maksimum dari elemen-elemen dalam jendela *pooling* tertentu, menonjolkan fitur yang paling kuat dalam wilayah tersebut. Di sisi lain, *average pooling* mengambil rata-rata dari nilai-nilai dalam jendela *pooling*, menghasilkan representasi yang lebih halus [16]. 3D *max pooling* merupakan ekstensi dari *max pooling* pada data dua dimensi ke data tiga dimensi. Pada 3D *max pooling*, operasi pooling dilakukan pada jendela kubus yang meliputi *depth*, *height*, dan *width* data masukan [16].

$$O(i,j,k) = \max_{d=0}^{F_d-1} \max_{h=0}^{F_h-1} \max_{w=0}^{F_w-1} I(S_d i + d, S_h j + h, S_w k + w) \tag{2.4}$$

Di mana	:
$O(i,j,k)$	: Nilai output pada posisi (i,j,k) setelah operasi max pooling 3D.
I	: Input yang akan diproses.
S_d, S_h, S_w	: *Stride* yang digunakan untuk pergeseran filter.
F_d, F_h, F_w	: Ukuran filter pooling pada dimensi.
i,j,k	: Posisi di output.
d,h,w	: Indeks dalam filter pooling untuk memilih elemen input.

Selain metode pooling konvensional seperti *max pooling* dan *average pooling*, terdapat juga teknik yang disebut *Global Average Pooling* (GAP). Berbeda dengan pooling biasa GAP langsung menghitung rata-rata dari seluruh peta fitur untuk setiap *channel*, yang menghasilkan satu nilai per *channel*. Teknik ini sering digunakan pada lapisan akhir jaringan, menggantikan lapisan *fully connected*. Keunggulan utamanya adalah mengurangi jumlah parameter secara signifikan, sehingga meminimalkan risiko *overfitting* dan meningkatkan

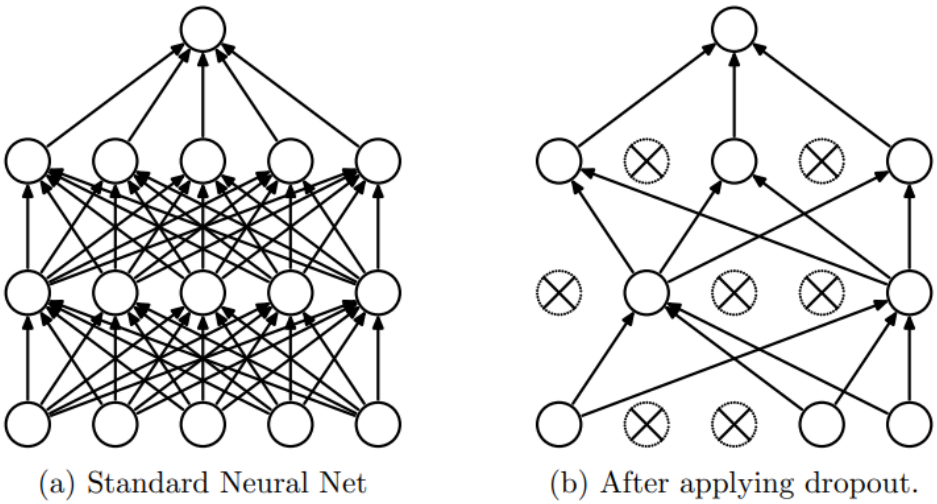
generalisasi model. Selain itu, GAP menghasilkan *output* yang lebih terfokus dan representatif terhadap kelas-kelas target [17].

$$O(c) = \frac{1}{D \times H \times W} \sum_{d=0}^{D-1} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} I(c,d,h,w) \tag{2.5}$$

Di mana	:
$O(c)$	: Nilai output Global Average Pooling untuk channel ke-c.
$I(c,d,h,w)$	: Nilai input pada channel c, posisi depth d, height h, dan width w.
D,H,W	: Ukuran dimensi spasial dan temporal dari peta fitur.
c	: Indeks channel dari peta fitur.

2.1.6.4 *Dropout Layer*

Dropout layer adalah teknik regularisasi yang digunakan dalam ANN untuk mencegah *overfitting* dengan cara menonaktifkan secara acak sebagian unit (*neuron*) selama proses pelatihan. *Dropout* bekerja dengan menonaktifkan setiap unit dalam lapisan dengan probabilitas tertentu (biasanya 0.5 untuk *hidden layer* dan 0.8 untuk lapisan input) selama setiap iterasi pelatihan, sehingga jaringan tidak bergantung terlalu banyak pada unit-unit tertentu [13]. Gambar 2.6 menunjukkan perbandingan *standard network* dengan *dropout network*. Pada jaringan dropout, beberapa unit dinonaktifkan secara acak selama pelatihan, menciptakan arsitektur yang lebih *robust*.



Gambar 2.6 Illustrasi *Dropout Layer* [19]

$$O(i) = \begin{cases} 0, & \text{jika } r_n > p \\ I(i) \times \frac{1}{p}, & \text{jika } r_n \leq p \end{cases} \quad (2.6)$$

Di mana	:
$O(i)$	: Matriks hasil pada posisi ke-i
$I(i)$	: Matriks input pada posisi ke-i
p	: Nilai probabilitas tetap
rn	: Angka acak yang dihasilkan dari rentang 0 sampai 1

2.1.6.5 *Fully Connected Layer*

Fully Connected layer adalah salah satu komponen penting dalam ANN, terutama pada bagian akhir arsitektur CNN. Lapisan ini menghubungkan setiap neuron di satu lapisan dengan setiap *neuron* di lapisan berikutnya, memungkinkan jaringan untuk mengintegrasikan semua fitur yang telah dipelajari dari lapisan sebelumnya. Lapisan ini dijelaskan sebagai bagian yang bertugas mengonversi representasi data yang kompleks menjadi *output* yang dapat digunakan, seperti skor probabilitas untuk klasifikasi. Persamaan 2.6 merupakan rumus keluaran dari lapisan *fully connected* [16].

$$y = f(Wx + b) \quad (2.7)$$

Di mana	:
W	: Matriks bobot yang menghubungkan neuron-neuron di dua lapisan
x	: Matriks input, seperti data citra
b	: bias, nilai tambahan untuk setiap neuron guna meningkatkan fleksibilitas model
f	: fungsi aktivasi K

Lapisan ini memiliki karakteristik unik, yaitu konektivitas penuh antara *neuron*, yang memungkinkan jaringan menangkap pola global dari data. Namun, karena semua *neuron* saling terhubung, lapisan ini cenderung membutuhkan jumlah parameter yang besar, sehingga lebih intensif secara komputasi dibandingkan lapisan konvolusi. Lapisan *fully connected* adalah tahap akhir yang penting dalam CNN, karena bertugas menerjemahkan fitur-fitur abstrak yang telah dipelajari menjadi hasil akhir yang dapat digunakan dalam berbagai aplikasi,

seperti pengenalan pola atau klasifikasi objek [16].

2.1.6.6 *Softmax Layer*

Softmax layer adalah lapisan output yang umum digunakan dalam ANN untuk masalah klasifikasi multi-kelas. Fungsi ini mengubah logits (nilai output sebelum aktivasi) menjadi distribusi probabilitas di mana setiap nilai berada dalam rentang [0, 1] dan jumlah seluruh nilai adalah 1. Dengan demikian, *softmax* memungkinkan model untuk memberikan prediksi yang dapat diinterpretasikan sebagai probabilitas suatu input termasuk dalam suatu kelas tertentu. Cara *softmax* untuk mendapatkan nilai probabilitas satu objek dijelaskan pada persamaan 2.8.

$$f_i(\vec{v}) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.8}$$

Di mana	:
$fi(\vec{v})$	: probabilitas objek ke-i
e	: bilangan euler bernilai 2.71828183
$\vec{v}$	: vektor yang bersisi nilai seluruh objek
n	: panjang $\vec{v}$
i	: posisi nilai sebuah objek pada $\vec{v}$

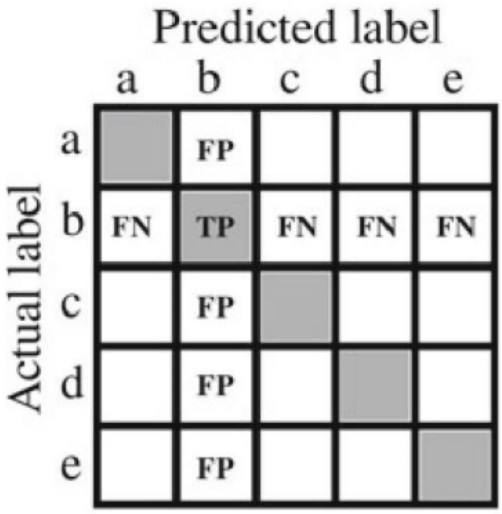
Lapisan *Softmax* digunakan sebagai lapisan terakhir dalam arsitektur CNN. Fungsinya adalah untuk menghasilkan probabilitas dari setiap kelas dan digunakan dalam proses evaluasi model. Di tahap ini, *softmax* membantu menghitung nilai loss, yaitu ukuran seberapa jauh prediksi model berbeda dari label sebenarnya. Nilai loss ini menjadi indikator akurasi model dalam mengklasifikasikan data. Perhitungan nilai *loss* dengan menggunakan fungsi *softmax* dapat dilihat pada Persamaan 2.9.

$$L_i = -\log(f_i(\vec{v})) \tag{2.9}$$

Di mana	:
L_i	: nilai loss
$f_i(\vec{v})$	: probabilitas objek ke-i

2.1.7 *Confusion Matrix*

Confusion Matrix merupakan salah satu metode evaluasi yang digunakan untuk menilai performa model klasifikasi, khususnya dalam kasus klasifikasi multi-kelas. Matriks ini berbentuk tabel dua dimensi yang menunjukkan jumlah prediksi yang dilakukan oleh model dibandingkan dengan label sebenarnya. Dapat dilihat pada gambar 2.7 bahwa dalam *confusion matrix* terdapat dua label yaitu *actual lable* dengan *predicted lable* [20].



Gambar 2.7 *Confusion Matrix* [20]

Confusion Matrix terdiri dari empat elemen utama, yaitu *True Positive* (TP), yaitu saat data sebenarnya positif dan diprediksi positif; *False Negative* (FN), yaitu ketika data sebenarnya positif namun diprediksi negatif; *False Positive* (FP), yaitu saat data sebenarnya negatif tetapi diprediksi positif; dan *True Negative* (TN), yaitu ketika data negatif dan diprediksi negatif. Keempat elemen ini digunakan untuk mengevaluasi kinerja model klasifikasi.

Accuraccy mengukur proporsi prediksi yang benar dari keseluruhan prediksi dan dihitung menggunakan rumus pada persamaan 2.10.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.10}$$

Precision mengukur sejauh mana hasil prediksi positif benar-benar relevan, dan dapat dihitung menggunakan rumus pada persamaan 2.11.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.11}$$

Recall menunjukkan seberapa banyak data positif yang berhasil diprediksi dengan benar oleh model, dapat dilihat pada persamaan 2.12.

$$\text{Recall} = \frac{TP}{TP+FN} \times 100\% \tag{2.12}$$

FMeasure menggambarkan keseimbangan antara presisi dan recall yang menghitung rata-rata harmonis dari keduanya. Rumus dari *FMeasure* dapat dilihat pada persamaan 2.13.

$$FMeasure = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.13}$$

2.1.8 Pustaka Pendukung

Dalam penelitian ini berbagai pustaka dan *library* digunakan untuk membangun, melatih, dan mengimplementasikan model *deep learning*. Berikut adalah pustaka utama yang digunakan beserta penjelasan dan daftar metode terkait:

2.1.8.1 TensorFlow

TensorFlow adalah pustaka perangkat lunak *open-source* untuk *machine learning* dan *deep learning* [25]. TensorFlow digunakan sebagai *backend* untuk Keras dalam membangun model 3D CNN dan mendukung operasi numerik pada tensor multidimensi. Tabel 2.1 menunjukkan daftar *method* yang digunakan dari pustaka TensorFlow.

Tabel 2.1 Daftar *method* yang digunakan dari pustaka TensorFlow

No.	*Method*	Masukan	Luaran	Keterangan

1.	`tf.keras.layers.Conv3D`	batch_size: int, depth: int, height: int, width: int, channels: int, kernel_size: tuple of 3 int, strides: tuple of 3 int	tf.Tensor: 5D tensor	Membuat lapisan konvolusi 3D untuk ekstraksi fitur dari data volumetrik atau spasial-temporal.
2.	`tf.keras.layers.MaxPooling3D`	batch_size: int, depth: int, height: int, width: int, channels: int	tf.Tensor: 5D tensor	Mengurangi dimensi data melalui *pooling* 3D.
3.	`tf.keras.layers.BatchNormalization`	batch_size: int, depth: int, height: int, width: int, channels: int	tf.Tensor: 5D tensor	Normalisasi batch untuk mempercepat pelatihan dan meningkatkan stabilitas model.
4.	`tf.keras.layers.GlobalAveragePooling3D`	batch_size: int, depth: int, height: int, width: int, channels: int	tf.Tensor: 2D	Mengurangi dimensi spasial (depth, height, width) dengan rata-rata global, menghasilkan tensor 2D sebelum masuk ke lapisan dense.

Pada CNN 3D, data input dan fitur yang dihasilkan selama proses konvolusi direpresentasikan sebagai tensor 5D, yaitu: *batch_size*: int, *depth*: int, *height*: int, *width*: int, *channels*: int.

2.1.8.2 Keras

Keras adalah pustaka tingkat tinggi yang berjalan di atas TensorFlow. Keras mempermudah pembangunan jaringan saraf tiruan dengan antarmuka modular dan ramah pengguna [26]. Model 3D CNN dibangun dengan menggunakan objek `Sequential` serta kombinasi dari `Conv3D`, `MaxPooling3D`, `BatchNormalization`, `GlobalAveragePooling3D`, `Dense`, dan `Dropout`. Tabel 2.2 menunjukkan daftar *method* yang digunakan dari pustaka Keras.

Tabel 2.2 Daftar *method* yang digunakan dari pustaka Keras

No.	***Method***	**Masukan**	**Luaran**	**Keterangan**

1.	`Sequential`	layers: list	Sequential	Menginisialisasi model *sequential*.
2.	`Dense`	batch_size: int, input_dim: int	batch_size: int, units: int	Menambahkan *fully connected layer* ke model.
3.	`Dropout`	batch_size: int, input_dim: int	batch_size: int, dropout_rate: float	Menerapkan regularisasi dropout untuk mencegah overfitting.
4.	`compile`	optimizer: str, loss: str, metrics: list	None	Mengkompilasi model dengan optimizer, fungsi loss, dan metrik evaluasi.
5.	`fit`	X: array, y: array, epochs: int, batch_size: int, validation_data: tuple	History	Melatih model pada data pelatihan dengan validasi.
6.	`predict`	X: array	array of predictions	Memprediksi kelas pada data input.

2.1.8.3 OpenCV

OpenCV adalah pustaka untuk pengolahan gambar dan video [27]. Dalam konteks 3D CNN, OpenCV digunakan terutama untuk preprocessing data seperti ekstraksi frame dari video, deteksi wajah, konversi warna, dan penskalaan citra. Tabel 2.3 menunjukkan daftar *method* yang digunakan dari pustaka OpenCV.

Tabel 2.3 Daftar *method* yang digunakan dari pustaka OpenCV

No.	***Method***	**Masukan**	**Luaran**	**Keterangan**
1.	`VideoCapture`	filename: str	objek Video Capture	Membaca file video dan memungkinkan ekstraksi frame.
2.	`cvtColor`	image: ndarray, code: int	ndarray	Mengonversi ruang warna gambar (misalnya BGR ke GRAY).

3.	`resize`	image: ndarray, dsize: tuple	ndarray	Mengubah ukuran gambar ke dimensi yang ditentukan.
4.	`Cascade Classifier`	xml_path: str	objek Cascade Classifier	Memuat classifier Haar untuk deteksi wajah.
5.	`detectMultiScale`	image: ndarray, scaleFactor: float, minNeighbors: int, minSize: tuple	array of rectangles	Mendeteksi wajah dalam gambar menggunakan Haar cascades, dimana scaleFactor menyesuaikan ukuran gambar, minNeighbors untuk jumlah tetangga minimal untuk deteksi objek, dan minSize ukuran minimal objek.

2.1.8.4 Scikit-Learn

Scikit-Learn digunakan untuk evaluasi model, termasuk pemisahan data, validasi silang, dan perhitungan metrik evaluasi seperti akurasi [28]. Tabel 2.4 menunjukkan daftar *method* yang digunakan dari pustaka Scikit-Learn.

Tabel 2.4 Daftar *method* yang digunakan dari pustaka Scikit-Learn

No.	***Method***	**Masukan**	**Luaran**	**Keterangan**
1.	`shuffle`	arrays: list of arrays	shuffled arrays	Mengacak urutan data untuk pelatihan yang lebih robust.
2.	`KFold`	n_splits: int, shuffle: bool, random_state: int	KFold object	Membuat objek untuk validasi silang k-fold.
3.	`accuracy_score`	y_true: array, y_pred: array	float	Menghitung akurasi dari prediksi model.
4.	`precision_score`	y_true: array, y_pred: array	float	Menghitung *precision* dari prediksi model.
5.	`recall_score`	y_true: array, y_pred: array	float	Menghitung *recall* dari prediksi model.

6.	`f1_score`	y_true: array, y_pred: array	float	Menghitung *f1 score* dari prediksi model.

2.2 Tinjauan Studi

Pada Tabel 2.5 akan dihabas perbandingan dari berbagai penelitian terkait metode deteksi kantuk pada pengemudi mobil:

Tabel 2.5 Tinjauan Studi

No	Peneliti	Judul	Rumusan Masalah	Hasil
1	M. I. B. Ahmed et al (2023)	*A Deep-Learning Approach to Driver Drowsiness Detection*	Mengembangkan pendekatan berbasis *deep learning* untuk deteksi kantuk dengan memanfaatkan CNN.	Model CNN menghasilkan akurasi sebesar 97%.
2	Sedik, A., Marey, M., dan Mostafa, H. (2023)	*An Adaptive Fatigue Detection System Based on 3D CNNs and Ensemble Models*	Mengusulkan sistem deteksi kantuk adaptif berbasis kombinasi 3D CNN dan *ensemble models* dengan augmentasi data menggunakan *Convolutional Generative Adversarial Network*.	Model 3D CNN mencapai akurasi 99% pada video, 98% pada gambar, dan 98% pada kombinasi dataset.

3	W. Xiang et al (2022)	*Driving Fatigue Detection Based on the Combination of Multi-Branch 3D-CNN and Attention Mechanism*	Mengusulkan sistem deteksi kantuk berbasis kombinasi 3D-CNN dan mekanisme atensi (SE module), dengan masukan berupa citra grayscale, gradien, dan optical flow dari wajah pengemudi.	Model 3D-SE-Net mencapai akurasi hingga 99% pada beberapa dataset (ZJU, NTHU, YawDD, dan FDF).
4	S. Chen, Z. Wang, and W. Chen (2021)	*Driver drowsiness estimation based on factorized bilinear feature fusion and a long-short-term recurrent convolutional network*	Deteksi kantuk pengemudi berbasis fitur tunggal dan fusi linear memiliki akurasi terbatas karena mengabaikan korelasi kompleks antar fitur dan informasi temporal.	Model hybrid CNN-LSTM dengan fusi bilinear non-linear (FBFF) mencapai akurasi 73.8%.
5	J.S. Wijinands et al. (2020)	*Real-time Monitoring of Driver Drowsiness on Mobile Platforms Using 3D Neural Networks*	Mengembangkan sistem deteksi kantuk berbasis 3D-CNN yang dirancang agar dapat berjalan pada perangkat *mobile* secara *real-time*, dengan memanfaatkan urutan frame video pengemudi.	Model 3D-CNN yang diusulkan mencapai akurasi 91,78% pada dataset NTHU-DDD.

Terdapat lima penelitian yang mengusulkan pendekatan berbeda dalam mendeteksi kantuk pengemudi mobil berbasis *computer vision*. Penelitian pertama dilakukan oleh Ahmed, M.I.B., et al., 2023 [1], yang mengusulkan pendekatan menggunakan CNN untuk mendeteksi kantuk. Pada tahap *preprocessing*, wajah

diekstraksi dari citra dengan menghapus latar belakang yang dianggap tidak relevan, kemudian ukuran gambar diubah menjadi 145x145 piksel untuk model CNN dan 64x64 piksel untuk model VGG16. *Dataset* kemudian diberi label yang terdiri dari kategori *open*, *closed*, *yawn*, dan *no-yawn*, yang diproses menggunakan teknik *one-hot encoding* dengan memetakan kategori *yawn* menjadi 0, *no-yawn* menjadi 1, *closed* menjadi 2, dan *open* menjadi 3. Data kemudian dibagi menjadi 70% untuk pelatihan dan 30% untuk pengujian model. Augmentasi data dilakukan untuk meningkatkan ketahanan model, diterapkan untuk memperbesar variasi gambar dengan cara merotasi pada berbagai sudut. Sistem ini mampu mencapai akurasi hingga 97% dalam mendeteksi kondisi mata tertutup dan aktivitas menguap. Namun, penelitian ini masih memiliki keterbatasan karena hanya menggunakan *public dataset* dengan asumsi wajah jelas tanpa kacamata.

Penelitian kedua dilakukan oleh Sedik, A., Marey, M., dan Mostafa, H., 2023 [7], yang mengusulkan sistem deteksi kantuk adaptif berbasis 3D CNN dan *ensemble models*. Pada tahap awal, fitur wajah dan mata diekstraksi menggunakan *Haar Cascaded Classifier* (HCC), kemudian data diperluas dengan teknik augmentasi berbasis *Conditional Generative Adversarial Network* (CGAN) untuk meningkatkan variasi citra. Model *machine learning* seperti SVM, KNN, *Random Forest*, dan MLP dibandingkan dengan pendekatan *deep learning* berbasis 2D CNN dan 3D CNN. Arsitektur 3D CNN yang diusulkan terdiri dari 16 lapisan, dengan kombinasi 3D *convolution*, *max pooling*, dan *global average pooling* untuk menangkap informasi spasial sekaligus temporal dari video. Sistem diuji pada *dataset* DROZY (video) serta *dataset* citra gejala kantuk (mata terbuka/tertutup dan menguap/tidak), dan juga pada kombinasi keduanya. Hasil eksperimen menunjukkan bahwa 3D CNN mencapai akurasi 99% pada video, 98% pada citra, serta 98% pada kombinasi *dataset*, melampaui akurasi penelitian sebelumnya yang hanya mencapai 97%. Dengan waktu pengujian rata-rata 25 ms per sampel, sistem ini berpotensi diimplementasikan secara *real-time* dalam sistem peringatan dini kendaraan. Namun, penelitian ini masih terbatas karena hanya diuji pada *dataset* tertentu dengan partisipan terbatas dan belum divalidasi pada kondisi nyata yang lebih beragam, seperti pengunaan aksesoris wajah, etnis dan usia.

Penelitian ketiga dilakukan oleh Xiang, W., et al., 2022 [6], mengusulkan sistem deteksi kantuk berbasis kombinasi *multi-branch* 3D CNN dan *attention mechanism* melalui *Squeeze-and-Excitation (SE) module*. Pada tahap *preprocessing*, wajah pengemudi dideteksi dan dilacak menggunakan pustaka Dlib, serta dilakukan strategi *frame aggregation* untuk mengatasi hilangnya deteksi

pada frame tertentu. Model ini menggunakan tiga cabang 3D-CNN yang masing-masing dilatih pada area mata kiri, mata kanan, dan mulut. Input berupa 10 *frame* video berturut-turut dengan lima kanal informasi (*grayscale*, *optical flow-x*, *optical flow-y*, *gradient-x*, *gradient-y*). Fitur yang diekstraksi kemudian dioptimalkan melalui SE module yang menyesuaikan bobot antar-kanal, sebelum diklasifikasikan menggunakan *full binary tree Support Vector Machine* (SVM). Penilaian kantuk dilakukan dengan menghitung nilai *Eye Aspect Ratio* (EAR) untuk mendeteksi kedipan dan *Mouth Aspect Ratio* (MAR) untuk mendeteksi aktivitas menguap. Hasil pengujian menunjukkan bahwa sistem ini mampu mencapai akurasi lebih dari 95% pada berbagai dataset (ZJU, NTHU-DDD, YawDD, dan FDF), serta terbukti lebih unggul dibandingkan metode konvensional seperti ResNet-50, MobileNetV3, dan EfficientNet. Namun, model ini masih memiliki keterbatasan, antara lain kompleksitas arsitektur yang tinggi sehingga memerlukan daya komputasi besar, risiko *overfitting* jika data latih kurang memadai, serta penurunan kinerja pada kondisi khusus seperti penggunaan kacamata hitam atau pencahayaan ekstrem.

Penelitian keempat dilakukan oleh Chen, S., et al., 2021 [4], yang mengembangkan pendekatan hybrid CNN-LSTM dengan faktorisasi bilinear untuk deteksi kantuk berbasis fitur visual mata dan mulut. Pada tahap *preprocessing*, wajah dideteksi menggunakan algoritma Viola-Jones, kemudian area mata dan mulut diekstraksi menggunakan dua model CNN terpisah (*SqueezeNet* untuk mulut dan VGG-16 untuk mata). Fitur-fitur ini digabungkan secara non-linear menggunakan metode *Factorized Bilinear Feature Fusion* (FBFF) sebelum diproses oleh LSTM untuk analisis temporal. Penelitian ini berhasil mencapai akurasi 73.8% pada dataset NTHU-DDD, mengungguli metode konvensional seperti DBN (61.9%) dan LRCN (62.9%). Namun, model ini masih memiliki keterbatasan, seperti kebutuhan komputasi yang relatif tinggi, pengujian yang terbatas pada satu *dataset*.

Penelitian kelima dilakukan oleh Wijnands, J.S., et al., 2020, [5] mengusulkan pendekatan deteksi kantuk berbasis *depthwise separable* 3D CNN yang dioptimalkan untuk berjalan secara *real-time* pada perangkat ponsel. Model ini dilatih dengan *transfer learning* menggunakan dataset ImageNet untuk fitur spasial dan Kinetics untuk fitur gerakan, lalu di-*fine-tune* pada *Driver Drowsiness Detection Dataset* (DDD). Hasilnya, sistem mencapai akurasi sebesar 73.9% dengan performa yang lebih baik pada kondisi sulit seperti pengemudi yang menggunakan kacamata hitam mencapai 76.8% dibandingkan dengan 59.3% untuk

metode berbasis 2D. Selain itu, model ini dapat berjalan secara *real-time* di ponsel *Android* dengan waktu inferensi sekitar 1.1 detik untuk 10 *frame* video, sehingga menunjukkan potensi besar untuk aplikasi praktis dalam meningkatkan keselamatan berkendara. Namun, model ini masih memiliki keterbatasan, seperti penggunaan *dataset* dengan jumlah subjek terbatas, serta kebutuhan komputasi yang masih cukup tinggi untuk perangkat dengan spesifikasi rendah.

Secara keseluruhan, kelima penelitian ini menunjukkan bahwa pendekatan berbasis *computer vision*, khususnya menggunakan CNN, LSTM dan 3D CNN, memberikan hasil yang menjanjikan dalam deteksi kantuk. Pada penelitian ini digunakan 3D CNN, di mana kombinasi teknik ini berpotensi menghasilkan sistem deteksi kantuk yang lebih baik, akurat, dan tahan terhadap variasi lingkungan.

2.3 Tinjauan Objek

Pada bab ini akan dijelaskan mengenai objek terkait dengan deteksi kantuk.

2.3.1 Orang Kantuk

Kantuk adalah kondisi yang terjadi ketika tubuh dan pikiran manusia mulai merasakan kelelahan, biasanya sebagai tanda bahwa individu membutuhkan istirahat atau tidur. Objek penelitian dalam hal ini adalah individu yang mengalami kantuk, yang dapat terlihat melalui berbagai tanda fisik dan perilaku. Kantuk pada manusia dapat memengaruhi tingkat kewaspadaan dan konsentrasi, yang berpotensi menurunkan performa dalam aktivitas sehari-hari, seperti berkendara, bekerja, atau belajar.

Orang yang mengalami kantuk sering kali menunjukkan gejala-gejala tertentu, seperti mata yang mulai terpejam, menguap, atau ekspresi wajah yang tampak lelah dan kurang fokus. Selain itu, perubahan perilaku seperti gerakan kepala yang tertunduk atau menurun, kesulitan dalam mempertahankan perhatian, serta kelambatan dalam reaksi juga merupakan indikator yang umum ditemui pada individu yang sedang mengantuk. Kondisi kantuk ini bisa disebabkan oleh berbagai faktor, seperti kurang tidur, monotoninya aktivitas yang dilakukan, atau bahkan faktor biologis yang mempengaruhi siklus tidur seseorang.

Kantuk terjadi karena tubuh membutuhkan pemulihan atau istirahat yang tidak dapat tercapai jika seseorang terus beraktivitas tanpa tidur yang cukup. Pada beberapa kasus, kantuk bisa berkembang menjadi tidur mikro, yaitu saat seseorang tertidur sejenak tanpa disadari dan tanpa kontrol. Kondisi ini berisiko tinggi, terutama jika terjadi pada saat seseorang sedang melakukan aktivitas yang

memerlukan kewaspadaan tinggi, seperti mengemudi atau bekerja di alat berat.

Gejala kantuk pada individu sangat bervariasi, tergantung pada tingkat kelelahan dan faktor penyebab lainnya. Berbagai gejala seperti penurunan waktu reaksi, gangguan kesadaran, atau terjadinya *microsleeps* (tertidur sejenak selama lebih dari 500 milidetik). Dalam konteks penelitian ini, pemahaman mengenai tanda-tanda fisik dan perilaku orang yang mengantuk sangat penting, karena ini menjadi dasar bagi pengembangan sistem yang dapat mendeteksi kondisi kantuk dengan lebih akurat.

2.3.2 *Dataset* Deteksi Kantuk

Dataset yang digunakan dalam penelitian ini adalah *University of Texas at Arlington Real-Life Drowsiness Dataset* (UTA-RLDD), sebuah himpunan data video yang dikembangkan secara khusus untuk mendeteksi kantuk dalam berbagai tingkatan, termasuk kondisi-kondisi awal yang sulit dikenali secara kasat mata. UTA-RLDD terdiri dari 180 video RGB dengan ukuran 1920x1080 dan 720x1280 berdurasi total sekitar 30 jam, yang direkam oleh 60 partisipan sehat dalam tiga kondisi tingkat kantuk yang berbeda:

1. *Alert* menggambarkan keadaan ketika seseorang sepenuhnya sadar dan memiliki tingkat kewaspadaan yang tinggi. Dalam keadaan ini, individu dapat berkonsentrasi dengan baik, bereaksi dengan cepat, dan mampu melakukan aktivitas seperti mengemudi dalam waktu lama tanpa mengalami rasa lelah atau kantuk.

2. *Low Vigilant* menunjukkan penurunan tingkat kewaspadaan. Pada tahap ini, tanda-tanda awal kantuk mulai muncul, meskipun belum terlalu mengganggu. Seseorang masih dapat melakukan aktivitas seperti mengemudi, namun fokus dan respons sudah tidak seoptimal kondisi waspada. Oleh karena itu, mengemudi dalam kondisi ini sebaiknya dihindari karena berpotensi menurunkan keselamatan.

3. *Drowsy* merupakan keadaan ketika rasa kantuk sudah cukup kuat sehingga seseorang harus berusaha keras untuk tetap terjaga. Konsentrasi dan respons menjadi sangat menurun, dan risiko tertidur secara tidak sadar meningkat. Dalam kondisi ini, melakukan aktivitas yang membutuhkan kewaspadaan tinggi, seperti mengemudi, sangat tidak disarankan karena berbahaya.

Masing-masing partisipan menyumbangkan tiga video, satu untuk setiap kondisi, dengan durasi sekitar 10 menit per video.

Seluruh video direkam secara mandiri oleh partisipan menggunakan kamera pribadi seperti ponsel atau *webcam*, dengan *frame rate* selalu di bawah 30 fps, mencerminkan kualitas perekaman kamera umum yang digunakan masyarakat. Partisipan diberikan kebebasan untuk merekam dalam rumah atau kampus, selama mereka merasa berada dalam kondisi sesuai dengan kategori yang ditentukan. Mereka juga diinstruksikan untuk menjaga konsistensi dalam sudut kamera, jarak, serta aktivitas selama perekaman video untuk setiap kondisi, agar menyerupai situasi di dalam kendaraan saat mengemudi. Posisi kamera yang direkomendasikan adalah sejajar wajah dan berada dalam satu lengan dari subjek, menyerupai posisi ponsel di dashboard mobil (lihat Gambar 2.8).

Komposisi demografis dalam *dataset* ini cukup beragam, mencakup rentang usia dari 20 hingga 59 tahun (rata-rata 25 tahun), serta berasal dari berbagai latar belakang etnis. Terdapat pula variasi dalam atribut fisik seperti penggunaan kacamata (21 dari 180 video) yang menambah tantangan dalam analisis visual.

Dataset ini disusun menjadi lima kelompok partisipan (fold) untuk keperluan validasi silang, yang memungkinkan evaluasi model secara menyeluruh. Dengan total ukuran data sebesar 111,3 GB dan variabilitas lingkungan, kondisi pencahayaan, kualitas video, serta perbedaan individu, UTA-RLDD menjadi salah satu *dataset* deteksi kantuk paling realistis dan komprehensif yang tersedia saat ini.



Gambar 2.8 Contoh Dataset [33]

Dataset ini sangat cocok digunakan untuk pelatihan model deteksi kantuk berbasis arsitektur 3D CNN, karena dapat membantu dalam mengenali pola mikro-ekspresi dan dinamika wajah yang kompleks yang terkait dengan tingkat kewaspadaan manusia dalam konteks nyata

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Pada Bab 1 telah dijelaskan bahwa kantuk saat berkendara merupakan salah satu faktor utama penyebab kecelakaan lalu lintas yang membahayakan keselamatan pengemudi. Beberapa tantangan dalam mendeteksi kantuk antara lain adalah keberagaman ciri fisik wajah pengemudi, serta kebutuhan sistem untuk memberikan hasil deteksi secara *real-time*. Selain itu, ketersediaan *dataset* yang berkualitas untuk melatih model juga menjadi kendala penting dalam pengembangan sistem deteksi yang handal.

Untuk mengatasi berbagai tantangan tersebut, penelitian ini menggunakan pendekatan 3D CNN, yang telah terbukti efektif dalam menganalisis data visual dan mengenali pola-pola yang kompleks. 3D CNN dipilih karena kemampuannya untuk mengekstraksi fitur waktu dari video. Dengan demikian, 3D CNN sangat sesuai untuk mendeteksi tanda-tanda kantuk berdasarkan pola visual seperti kondisi mata.

Penelitian ini terbagi dalam dua tahapan utama, yaitu tahap *training* dan tahap *testing*. Pada tahap *training*, model 3D CNN akan dilatih menggunakan *dataset* video pengemudi yang mencakup berbagai kondisi, seperti mata berkedip dan mata yang terpejam lama. Kemudian tahap *testing* yang bertujuan untuk mengevaluasi performa model dengan data yang belum pernah dilihat sebelumnya, untuk memastikan kemampuan model dalam melakukan generalisasi.

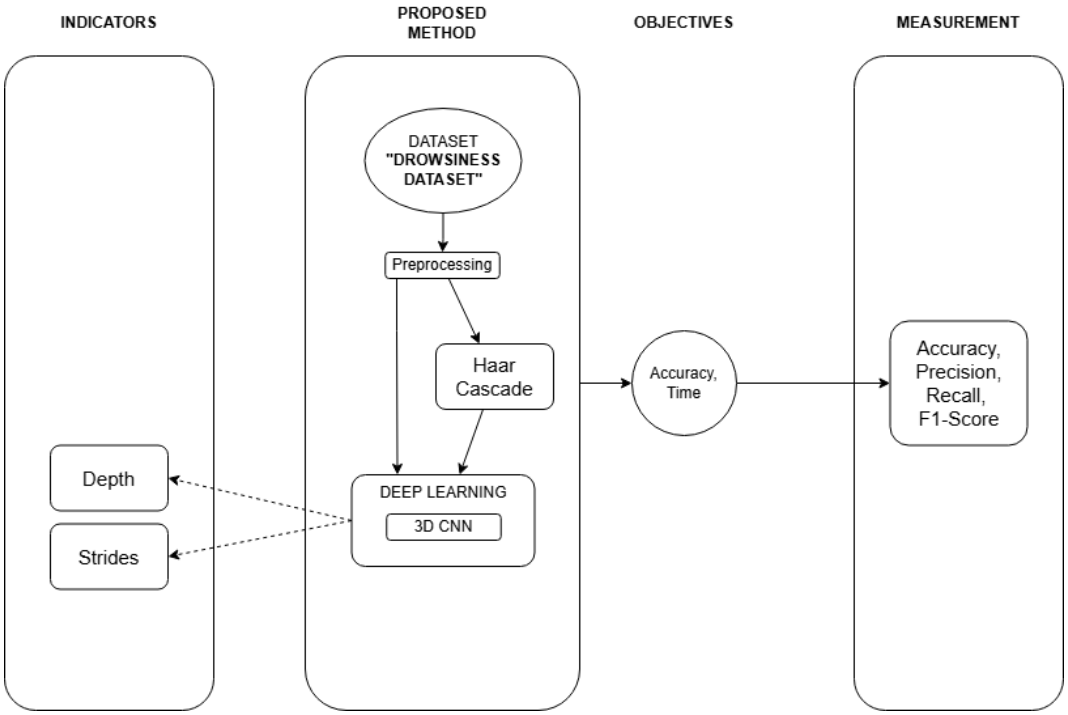
Input untuk model 3D CNN dalam penelitian ini adalah video wajah orang yang memuat kondisi kantuk dan tidak kantuk. *Output* dari sistem ini berupa prediksi tingkat kantuk pengemudi, yang selanjutnya dapat digunakan untuk memberikan peringatan dini jika kondisi kantuk terdeteksi. Hasil evaluasi pada tahap pengujian akan mencakup metrik seperti *accuracy*, *precision*, *recall*, dan f1-*score* yang menggambarkan seberapa baik model dapat mengenali tanda-tanda kantuk secara konsisten dan dalam waktu nyata.

Penelitian ini diharapkan dapat menghasilkan sistem deteksi kantuk berbasis 3D CNN yang tidak hanya akurat, tetapi juga dapat diterapkan dalam situasi dunia nyata untuk meningkatkan keselamatan dalam berkendara. Dengan mengatasi berbagai tantangan yang ada, penelitian ini berpotensi memberikan kontribusi signifikan terhadap pengembangan teknologi keselamatan pada

kendaraan modern.

3.2 Kerangka Pemikiran

Bagian ini akan menjelaskan kerangka pemikiran dari metode yang diusulkan untuk melakukan deteksi kantuk pada pengemudi mobil dengan metode 3D CNN.



Gambar 3.1 Kerangka Pemikiran

Gambar 3.1 menggambarkan kerangka pemikiran yang digunakan dalam penelitian ini. Berikut adalah penjelasan mengenai setiap komponen yang ada dalam gambar 3.1:

1. *Indicators* adalah parameter-parameter yang berperan penting untuk menentukan seberapa baik model 3D CNN dapat berfungsi dan menghasilkan *output* yang diinginkan. Dalam penelitian ini, indikator dianalisis secara berkala selama proses pelatihan dan pengujian untuk memastikan kinerja model yang optimal. Beberapa indikator yang digunakan antara lain:

   (a) *Depth*: mengacu pada jumlah *frame* berurutan yang digunakan sebagai satu input dalam jaringan CNN 3D. Parameter ini sangat penting karena memengaruhi seberapa baik model dapat menangkap informasi spasial dan temporal secara bersamaan. Jumlah frame yang terlalu

sedikit mungkin tidak cukup untuk mengenali pola pergerakan seperti kedipan atau menguap, sedangkan terlalu banyak dapat menyebabkan peningkatan beban komputasi dan risiko *redundant information*. Pada penelitian ini digunakan panjang urutan frame sebesar 32, dan 64 frame untuk menemukan titik optimal pada berbagai task video recognition [23]

(b) *Strides*: strides mengacu pada jarak antar posisi filter yang diterapkan pada input data, baik itu dalam dimensi spasial (ruang) maupun temporal (waktu). Dalam kasus pemrosesan video, strides mengatur seberapa sering model memindahkan filter untuk mengambil sampel dari frame video. Jika nilai strides kecil, seperti 1, model akan memproses setiap frame secara lebih terperinci, memungkinkan deteksi gerakan halus dengan akurasi tinggi, namun membutuhkan lebih banyak waktu komputasi. Sebaliknya, jika nilai strides lebih besar, seperti 2 atau 3, model akan memproses frame dengan interval yang lebih lebar, yang dapat mempercepat inferensi tetapi berisiko kehilangan informasi temporal yang penting. Pada penelitian ini, digunakan strides sebesar 1 dan 2 [23].

2. *Proposed method* adalah proses yang diusulkan oleh penelitian ini. Dataset diambil dari google sites yang bernama UTA-RLDD. Kemudian dataset di preprocessing menjadi ukuran 128x128 dengan 30 *frame per second* (fps) sesuai dengan ukuran input pada 3D CNN. Setelah itu dataset dimasukkan ke dalam *haar cascade* untuk mengekstrak wajah. *Haar Cascades* merupakan metode deteksi objek yang efisien dan cepat, khususnya untuk mendeteksi wajah dan mata secara *real-time* dalam pengolahan citra. Metode ini bekerja berdasarkan ekstraksi fitur *Haar-like*. Setiap fitur *Haar-like* dihitung sebagai selisih intensitas piksel antara area putih dan hitam dalam persegi panjang. Metode ini bekerja berdasarkan ekstraksi fitur *Haar-like*. Setiap fitur *Haar-like* dihitung sebagai selisih intensitas piksel antara area putih dan hitam dalam persegi panjang. Pada penelitian ini membandingkan penggunaan *haar cascade* dengan tidak menggunakkan *haar cascade* [10]. Kemudian 3D CNN digunakan sebagai metode utama untuk menganalisis data video dan mendeteksi pola yang berkaitan dengan tanda-tanda kantuk seperti mata terpenjam, 3D CNN dipilih karena kemampuannya dalam mengenali fitur-fitur penting dalam citra melalui lapisan-lapisan konvolusi. 3D CNN pada penilitian ini menggunakan

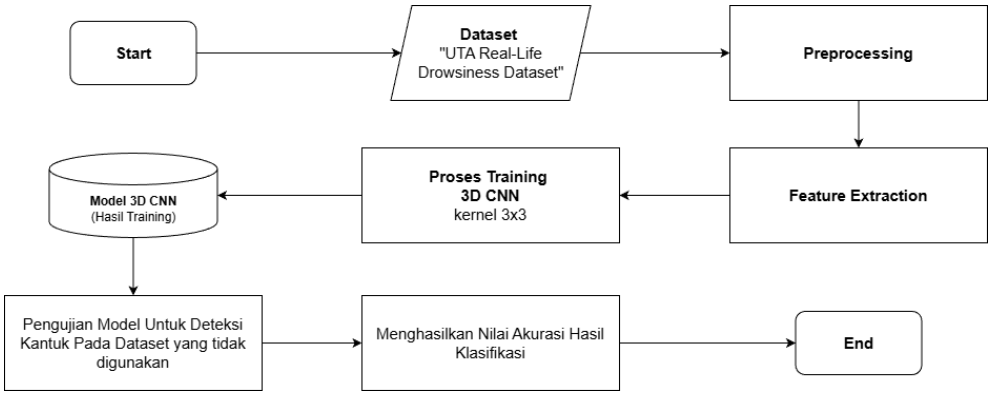
learning rate: 0.001 *batch size*: 8, dan *epochs*: 40.

3. *Objectives* adalah tujuan utama dari penelitian ini. Dalam hal ini, fokus penelitian untuk mengembangkan model yang mampu mengenali tanda-tanda kantuk dengan tingkat akurasi yang tinggi dan waktu yang cepat. Keberhasilan model ini diukur berdasarkan tingkat akurasi prediksi yang dihasilkan terhadap data pengujian.

4. *Measurement* adalah metrik pengukuran untuk menguji model yang dibangun pada penelitian ini. Dalam penelitian metrik pengukuran yang digunakan adalah *accuracy*, *precision*, *recall*, *f1-score* untuk mengukur akurasi model.

3.3 Urutan Proses Global

Bagian ini akan menjelaskan urutan proses global dari metode yang diusulkan untuk melakukan deteksi kantuk pada pengemudi mobil dengan metode CNN.



Gambar 3.2 Urutan Proses Global

Gambar 3.2 menggambarkan proses penelitian secara global untuk mendeteksi kantuk pada pengemudi mobil menggunakan metode 3D CNN. Proses dimulai dengan penggunaan dataset ”UTA Real Life Drowsiness Dataset” yang berisi video kondisi kantuk dan tidak kantuk. *Dataset* ini dipilih dan dipersiapkan untuk melatih model deteksi kantuk berbasis *computer vision*.

Kemudian pada tahap *preprocessing* dataset di *resize* menggunakan pustaka OpenCV yang bernama *resize* 2.3 menjadi ukuran 128x128 sesuai dengan ukuran masukkan untuk arsitektur 3D CNN. Hal ini dilakukan untuk mengurangi dimensi data input, sehingga menghemat komputasi dan memastikan konsistensi input ke 3D CNN. Dataset juga diekstraksi menggunakan *haar cascade* untuk mengambil muka dari setiap *frame* video pada tahap *feature extraction*. *Haar*

cascade digunakan untuk efisiensi komputasi, meningkatkan akurasi, dan kecepatannya untuk memproses real-time.

Tahap berikutnya adalah proses *training* 3D CNN, di mana model 3D CNN dilatih menggunakan data pelatihan untuk mempelajari pola kantuk, seperti kondisi mata, dan menguap. Setelah pelatihan selesai, model yang telah dilatih diuji dengan data yang belum pernah dilihat sebelumnya pada tahap Pengujian Model. Tahap ini bertujuan untuk menilai kinerja model dalam mendeteksi kantuk pada *dataset* uji dengan menggunakan metrik evaluasi akurasi. Pada akhirnya, penelitian ini menghasilkan model 3D CNN yang mampu mendeteksi kantuk.

3.4 Analisis Manual

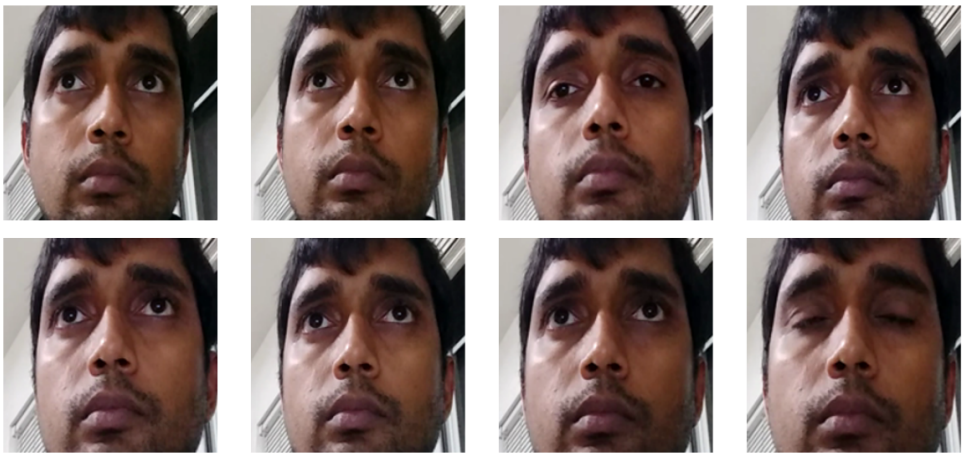
Pada bagian ini, dilakukan analisis melalui perhitungan secara manual. Tujuan dari analisis ini adalah untuk memberikan pemahaman yang lebih mendalam mengenai proses-proses yang terlibat dalam penelitian ini.

3.4.1 *Data Sampling*

Pada bagian ini dilakukan analisis manual terhadap *dataset* yang digunakan dalam penelitian ini. Dataset yang digunakan berasal dari *google sites* yang bernama *UTA-RLDD*. Dataset ini terdiri dari 180 video yang berukurang 1920x1080 dan 720x1280 dengan 3 kelas yaitu *alert*, *low vigilant*, dan *drowsy*. Setiap video berdurasi sekitar 10 menit. Namun, untuk keperluan deteksi kantuk menggunakan arsitektur 3D CNN, video dibagi menjadi 2 kelas yaitu *drowsy* dan *not drowsy*, kemudian diubah segmen-segmen pendek (klip) berdurasi 5 detik dimana setiap detik 30 fps. Pada tahap ini segmen pendek berdurasi 5 detik untuk *drowsy* dipilih secara manual, dimana dari video 10 menit dilihat dari kondisi kantuk seperti mata berkedip yang lebih lama dan mulut yang menguap, sedangkan untuk *not drowsy* dipilih secara acak dari *dataset alert*. Dapat dilihat pada gambar 3.3 dan gambar 3.4 untuk contoh dari kondisi yang menunjukkan kantuk dan tidak kantuk.



Gambar 3.3 Contoh *frame drowsy*

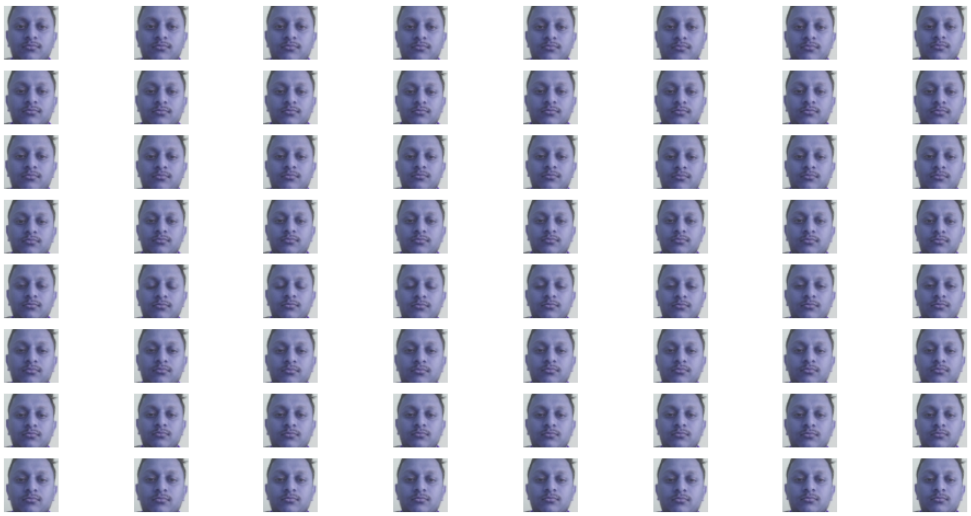


Gambar 3.4 Contoh *frame not drowsy*

Pada penilitian ini digunakan 32 *depth* dan 64 *depth* sehingga diambil total 1.067 detik untuk 32 *depth* dan 2.134 detik untuk 64 *depth*. Hal ini dilakukan untuk mengurangi beban komputasi serta menyesuaikan dengan cara kerja 3D CNN yang mengolah data dalam bentuk *spatio-temporal blocks*. Untuk menangkap dinamika temporal secara lebih efektif dari klip-klip 5 detik tersebut, diterapkan teknik *sliding window* pada tahap *preprocessing*. *Sliding window* melibatkan pembagian klip menjadi *sequence frame* bertahap dengan ukuran 32 atau 64 frame sesuai depth yang digunakan, di mana window bergeser dengan *stride* 4 untuk menghasilkan *overlap* antar *sequence frame* [24]. Pendekatan ini memungkinkan model 3D CNN untuk mengekstrak fitur *spatio-temporal* tanpa kehilangan informasi transisi antar-*frame*, seperti perubahan gerakan mata atau kepala yang bertahap, sehingga meningkatkan akurasi deteksi kantuk sambil menjaga efisiensi komputasi.

Kemudian, dataset yang telah dibagi ke dalam klip-klip pendek antara 32

dan 64 tersebut dimasukkan kedalam *haar cascade* untuk proses mendeteksi *region of interest* (ROI) muka yang dapat dliihat proses haar cascade pada algoritme 3.1. Setiap klip video kemudian diproses secara *frame-by-frame*, di mana setiap *frame* terlebih dahulu dikonversi ke dalam citra grayscale untuk dapat dideteksi oleh *haar cascade*. Selanjutnya, metode *Haar Cascade* digunakan untuk mendeteksi area wajah pada setiap *frame*. Apabila wajah berhasil terdeteksi, maka dilakukan proses slicing untuk mengambil bagian *frame* yang merupakan ROI wajah berdasarkan koordinat hasil deteksi. ROI wajah tersebut kemudian diubah ukurannya menjadi dimensi yang seragam agar konsisten sebagai masukan model. Sebaliknya, jika pada suatu *frame* tidak terdeteksi wajah, maka *frame* tersebut digantikan dengan citra kosong. Proses ini dilakukan hingga jumlah *frame* dalam setiap klip mencapai batas yang ditentukan, sehingga seluruh klip memiliki panjang dan format data yang seragam untuk tahap pemrosesan selanjutnya. Contoh hasil dari *haar cascade* dapat dilihat pada gambar 3.5.



Gambar 3.5 Hasil *haar cascade*

Algorithm 3.1 Extract faces untuk Depth=64

1: Memuat model deteksi wajah *face_cascade* menggunakan *CascadeClassifier('haarcascade_frontalface_default.xml')* pada tabel 2.3.
2: Membaca file input video dengan menggunakan *VideoCapture('/content/drive/MyDrive/UTA-RLDD')* pada tabel 2.3.
3: Konversi frame dari video ke grayscale menggunakan *cvtColor(image, cv2.COLOR_BGR2GRAY* pada tabel 2.3.
4: Mendeteksi wajah dalam frame menggunakan *detectMultiScale(image, scaleFactor=1.1, minNeighbors=5, minSize(30, 30)* pada tabel 2.3.
5: *Resize* frame sesuai dengan ukuran *frame size* yang diinginkan (frame_size=128,128) menggunakan *resize* pada tabel 2.3.
6: Ulangi langkah ke-3 hingga jumlah frame mengikuti ukuran *max_frames*=64. Jika jumlah frame video tidak mencukupi maka frame akan diisi dengan frame kosong, dimana frame kosong adalah array yang berisi 0 dengan ukuran yang sama dan tipe yang sama.

Pada algoritme 3.1 jika memakai depth=32 maka *max_frames* diganti menjadi 32. Dataset juga di *preprocessing* menggunakan *resize* menjadi ukuran 128x128, teknik *resize* yang digunakan pada tahapan ini adalah *downsampling*, contoh *resize* dapat dilihat pada gambar 3.6. *Resize* menjadi 128x128 dilakukan karena faktor keterbatasan sumber daya sehingga diperlukan untuk melakukan *resize*. Hal ini juga dilakukan untuk mengurangi dimensi data input, sehingga menghemat komputasi, mencegah *overfitting* pada model CNN, dan memastikan konsistensi input ke 3D CNN.



Gambar 3.6 Contoh *resize* citra

Selanjutnya dataset diacak lalu untuk mengurangi bias dan meningkatkan reliabilitas digunakan *k-fold cross-validation* dengan k sebesar 10 [7] dimana pada *k-fold cross-validation* terdapat dua subset yaitu subset *training* dan *subset testing*. Subset *training* digunakan untuk melatih model 3D CNN agar mampu mempelajari pola-pola *spatio-temporal* yang membedakan kondisi pengemudi

yang sadar dan mengantuk, sedangkan subset *testing* digunakan untuk mengukur kinerja generalisasi model terhadap data yang belum pernah dilihat sebelumnya. Proses ini dapat dilihat pada algoritme 3.2.

Algorithm 3.2 Training dan testing model

1: Inisialisasi *KFold(n_splits=10, shuffle=True, random_state=42)* menggunakan library pada tabel 2.4.
2: Inisialisasi model dengan memanggil method *build_3d_cnn*.
3: Latih model menggunakan *model.fit(X_train, y_train, epochs=40, batch_size=8, validation_data=(X_val, y_val), callbacks)* menggunakan *fit* pada tabel 2.2.
4: Prediksi kelas pada data input menggunakan *model.predict(X_val)* pada tabel 2.2.
5: Hitung akurasi, precision, recall dan f1 score model berdasarkan label sebenarnya dan yang diprediksi menggunakan *accuracy_score(y_val, val_predictions) precision_score(y_val, val_predictions) recall_score(y_val, val_predictions) f1_score(y_val, val_predictions)* sesuai pada tabel 2.4.

Analisis ini memberikan gambaran bahwa dataset yang digunakan sudah memenuhi kriteria yang baik untuk pengembangan model 3D CNN, baik dari segi kualitas maupun kuantitas data.

3.4.2 *3D Convolutional Neural Network*

Arsitektur 3D CNN yang digunakan dapat dilihat pada Tabel 3.1 [7]. Pada arsitektur ini digunakan padding *valid*, yaitu jenis padding yang tidak menambahkan piksel apa pun di bagian tepi input, sehingga ukuran keluaran akan mengecil setelah proses konvolusi. Pendekatan ini membantu model memfokuskan ekstraksi pada fitur-fitur lokal yang benar-benar terdeteksi dalam data tanpa pengaruh tepi buatan. Sebagai pembanding, terdapat juga *padding same*, yaitu teknik yang menambahkan piksel di tepi input sehingga ukuran keluaran tetap sama dengan ukuran masukan. *Padding same* biasanya digunakan ketika informasi di bagian tepi data perlu dipertahankan atau ketika konsistensi resolusi pada tiap layer diperlukan.

Selain itu, arsitektur ini menggunakan *stride* berukuran 1 pada tahap awal untuk mempertahankan resolusi spasial sehingga fitur-fitur penting dapat diekstraksi secara lebih rinci. Pada lapisan berikutnya digunakan *stride* berukuran 2 untuk mengurangi ukuran data menjadi setengahnya, sehingga komputasi menjadi lebih efisien serta mempermudah model dalam menangkap fitur-fitur yang lebih dominan.

Algorithm 3.3 Build_3D_CNN untuk Strides=2

Require: *Depth*: Jumlah masukkan frame ke dalam 3D CNN.
Require: *Strides*: Jarak perpindahan antara frame video.

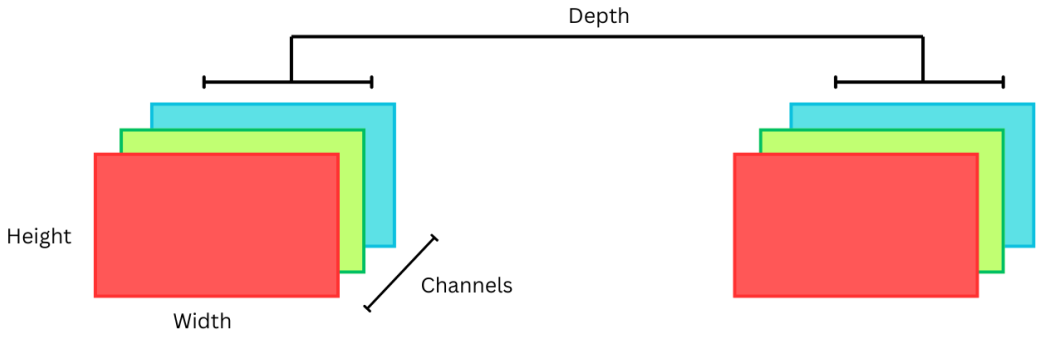
1: Inisialisasi model dengan menggunakan *models.Sequential* pada tabel 2.2.
2: Menambahkan layer konvolusi menggunakan *Conv3D(filter, kernel_size=(3,3,3), activation='relu', padding='valid'* pada tabel 2.1.
3: Menambahkan layer max pooling menggunakan *MaxPooling3D(pool_size=(2,2,2), stride=(2,2,2), padding='same'* pada tabel 2.1.
4: Menambahkan layer batch normalization menggunakan *BatchNormalization* pada tabel 2.1.
5: Ulangi langkah ke-2 sampai ke-4 hingga jumlah layer mencapai 4.
6: Menambahkan layer global average pooling menggunakan *GlobalAveragePooling3D* pada tabel 2.1.
7: Menambahkan layer dense dengan 512 unit dan *ReLU*, layer dropout dengan nilai 0.5, dan layer dense dengan 2 unit dan *Softmax* pada tabel 2.2.
8: Compile model menggunakan *model.compile* pada tabel 2.2.

Dapat dilihat pada algoritme 3.3 untuk alur yang digunakan dalam membuat model 3D CNN. Proses Training ini digunakan untuk stride yang berjumlah 2, dan untuk stride 1 maka ganti menjadi stride=(1,2,2).

Tabel 3.1 Arsitektur *3D-CNN* Untuk Depth=64

No.	Layer	Input Dimension	Kernel Size	Output Dimension	Keterangan
1.	Convolutional Layer 1	(128, 128, 64, 3)	3×3×3×32	(126, 126, 62, 32)	stride 1, padding 'valid', ReLU.
2.	Max Pooling Layer 1	(126, 126, 62, 32)	-	(63, 63, 31, 32)	Pooling 2×2×2, stride 2.
3.	Batch Normalization 1	(63, 63, 31, 32)	-	(63, 63, 31, 32)	Normalisasi batch.
4.	Convolutional Layer 2	(63, 63, 31, 32)	3×3×3×64	(61, 61, 29, 64)	stride 1, padding 'valid', ReLU.
5.	Max Pooling Layer 2	(61, 61, 29, 64)	-	(30, 30, 14, 64)	Pooling 2×2×2, stride 2.

6.	Batch Normalization 2	(30, 30, 14, 64)	-	(30, 30, 14, 64)	Normalisasi batch.
7.	Convolutional Layer 3	(30, 30, 14, 64)	3×3×3×64	(28, 28, 12, 64)	stride 1, padding 'valid', ReLU.
8.	Max Pooling Layer 3	(28, 28, 12, 64)	-	(14, 14, 6, 64)	Pooling 2×2×2, stride 2.
9.	Batch Normalization 3	(14, 14, 6, 64)	-	(14, 14, 6, 64)	Normalisasi batch.
10.	Convolutional Layer 4	(14, 14, 6, 64)	3×3×3×128	(12, 12, 4, 128)	stride 1, padding 'valid', ReLU.
11.	Max Pooling Layer 4	(12, 12, 4, 128)	-	(6, 6, 2, 128)	Pooling 2×2×2, stride 2.
12.	Batch Normalization 4	(6, 6, 2, 128)	-	(6, 6, 2, 128)	Normalisasi batch.
13.	Global Average Pooling	(6, 6, 2, 128)	-	(128)	Rata-rata fitur dari setiap map (128 channel).
14.	Fully Connected (Dense 1)	(128)	-	(512)	Dense layer, 512 neuron, aktivasi ReLU.
15.	Dropout	(512)	-	(512)	Dropout untuk mencegah overfitting.
16.	Output Layer (Dense 2)	(512)	-	(2)	Dense layer, 2 neuron (klasifikasi biner, Softmax).



Gambar 3.7 Contoh Array video

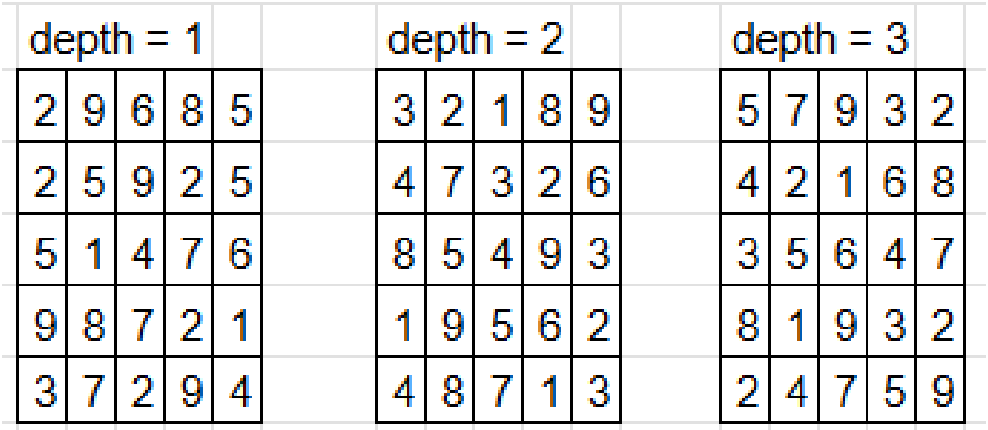
Gambar 3.7 merupakan sebuah ilustrasi dari sebuah video yang akan dimasukkan ke dalam masukan pada tahap CNN. Di mana pada gambar 3.7 terdapat 4 atribut yaitu:

1. *Depth*: menggambarkan dimensi temporal. Jika feature map merupakan data video, maka *depth* menunjukkan jumlah *frame* dalam video tersebut. Sementara untuk citra, nilai depthnya adalah 1. Sehingga untuk video dimensi ini merepresentasikan aspek waktu dari data.

2. *Height*: menyatakan ukuran *vertikal* dari *frame*, yang merupakan salah satu dimensi spasial.

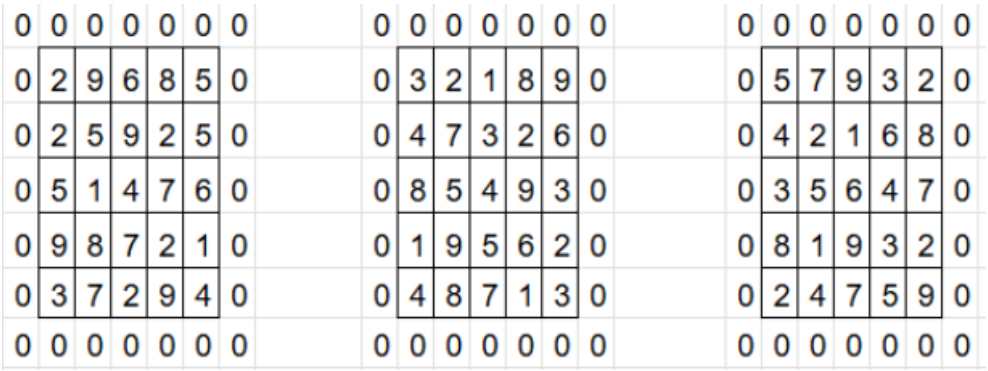
3. *Width*: menyatakan ukuran *horizontal* dari *frame*, yang merupakan salah satu dimensi spasial.

4. *Channel*: merepresentasikan jumlah *channel* warna pada setiap *frame*, seperti *channel red*, *green*, dan *blue* (RGB). Dimensi ini menunjukkan informasi warna dalam data.



Gambar 3.8 Contoh Array video

Gambar 3.8 menunjukkan contoh matriks video pada CNN yang mempresentasikan piksel pada video sebagai masukan pada tahap CNN. Pada analisis manual ini digunakan matriks berukuran 5x5x3 untuk mempermudah perhitungan konvolusi. Pada array video dilakukan *padding='same'* yang menambahkan angka 0 di sekitar array video, dapat dilihat pada gambar 3.9 untuk hasil dari *padding='same'*.



Gambar 3.9 Contoh Padding Pada Array video

Kernel pada penelitian ini berukuran 3x3×3x2 dengan input video sebesar 126x126x3 piksel. Arsitektur model terdiri atas 4 lapisan konvolusi yang masing-masing memiliki 32, 64, 64, dan 128 filter. Gambar 3.10 menunjukkan contoh array kernel yang digunakan. Perhitungan kernel menggunakan *Standard Normal Distribution* dapat dilihat pada 3.2 yang menampilkan contoh perhitungan manual kernel. Proses perhitungan ini bertujuan untuk meminimalkan kesalahan model selama pelatihan dan menghindari terjadinya *overfitting*.

0.12734	0.25381	-0.08952	0.17589	-0.31243	0.08912	0.09347	-0.14623	0.07934
-0.18377	-0.31429	-0.09863	-0.24756	0.18542	-0.05371	0.16289	-0.28539	0.09761
-0.05196	0.22873	0.14411	0.09672	-0.18562	0.12534	-0.15829	0.22154	-0.08429
-0.12953	0.21178	-0.08539	0.17634	-0.26347	0.14982	-0.05674	0.09391	-0.12841
0.14562	-0.32247	0.09112	-0.09834	0.18752	-0.09274	0.07641	-0.20632	0.18425
-0.03972	0.25834	-0.17261	0.06781	-0.21176	0.12491	-0.15268	0.30145	-0.11234

Gambar 3.10 Contoh Array Kernel 3x3x3x2

$$
\begin{aligned}
k &= \frac{1}{\sigma \times \sqrt{2\pi}} \times e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\
&= \frac{1}{2 \times \sqrt{2 \times 3.14}} \times 2.7182818^{-\frac{(7-5)^2}{2\times 2^2}} \\
&\approx 0.12098536
\end{aligned}
$$

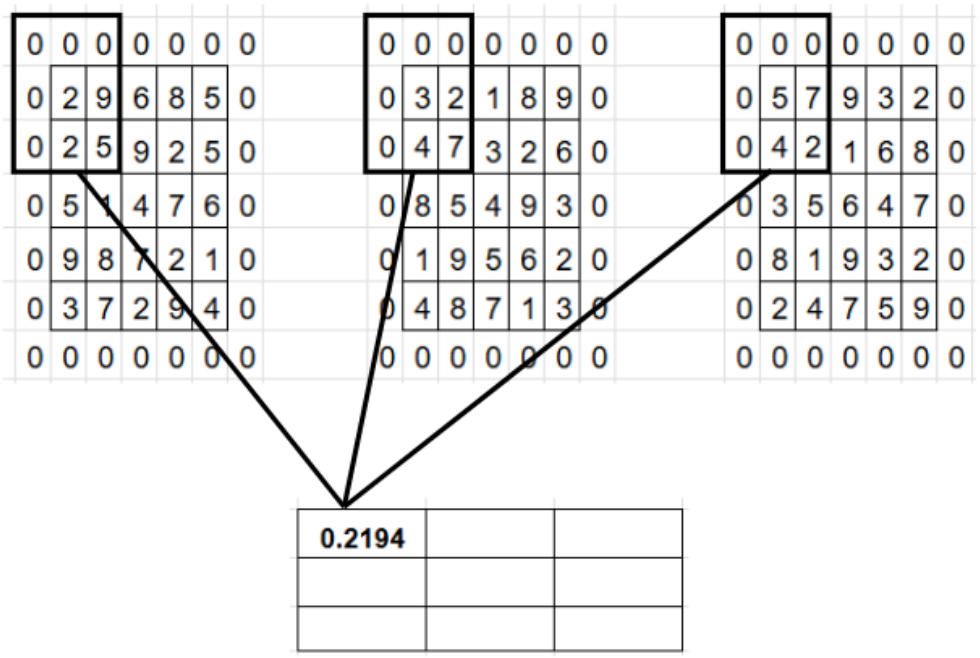
3.2 Perhitungan *Standard Normal Distribution*

Dalam proses konvolusi, nilai-nilai pada citra input akan diproses menggunakan kernel. Pada analisis manual ini digunakan kernel berukuran 3×3×3×2 seperti yang ditunjukkan pada Gambar 3.10. Contoh perhitungannya dapat dilihat pada 3.3 yang menggunakan rumus 2.2. Perhitungan 3.3 menggambarkan perhitungan *dot product* pada posisi (0,0,0). Operasi ini dimulai dari posisi awal matriks, yaitu indeks [0,0] dan pada 3 *depth* karena kernelnya berukuran 3x3x3, kemudian bergeser satu per satu hingga mencapai titik akhir di sudut kanan bawah, yakni indeks [x,y] dan dilakukan pada setiap frame. Setiap langkah konvolusi dilakukan melalui operasi perkalian antara elemen citra dan kernel (I×K), yang kemudian menghasilkan representasi baru berupa *feature map*. Seluruh proses ini bertujuan untuk mengekstraksi fitur dari citra.

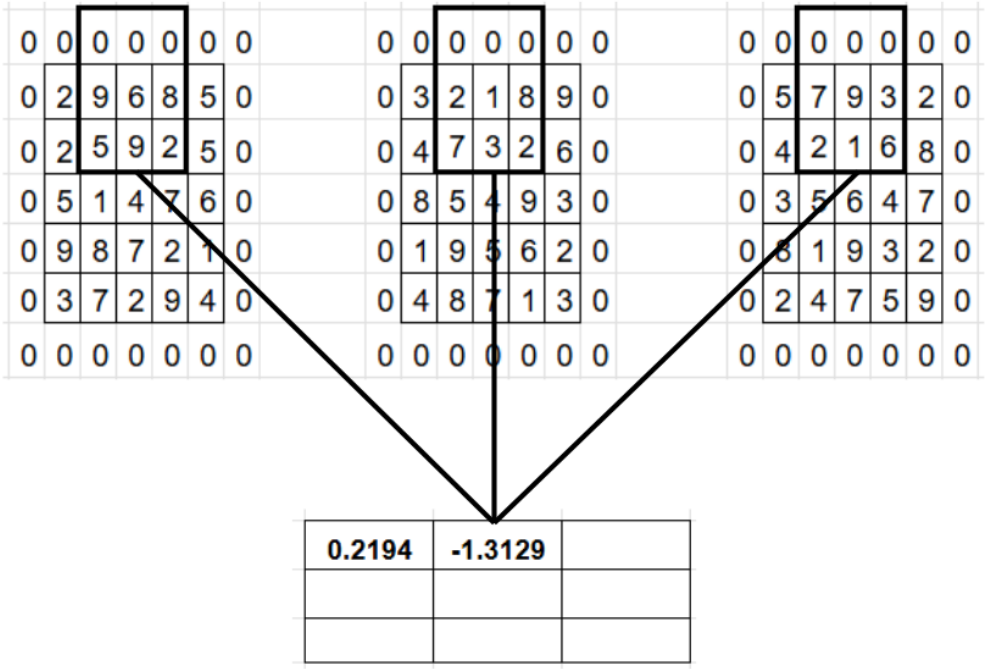
3.3 Perhitungan *feature map* 3D

$$
\begin{aligned}
O(0,0,0) = {} & \underbrace{I(0,0,0)\cdot K(0,0,0)}_{0\times 0.12734} + \underbrace{I(0,0,2)\cdot K(0,0,1)}_{0\times 0.25381} + \underbrace{I(0,0,2)\cdot K(0,0,2)}_{0\times(-0.08952)} \\
& + \underbrace{I(0,1,0)\cdot K(0,1,0)}_{0\times(-0.18377)} + \underbrace{I(0,1,1)\cdot K(0,1,1)}_{2\times(-0.31429)} + \underbrace{I(0,1,2)\cdot K(0,1,2)}_{9\times(-0.09863)} \\
& + \underbrace{I(0,2,0)\cdot K(0,2,0)}_{0\times(-0.05196)} + \underbrace{I(0,2,1)\cdot K(0,2,1)}_{2\times 0.22873} + \underbrace{I(0,2,2)\cdot K(0,2,2)}_{5\times 0.14411} \\
& + \underbrace{I(0,0,0)\cdot K(0,0,0)}_{0\times 0.17589} + \underbrace{I(0,0,1)\cdot K(0,0,1)}_{0\times -0.31243} + \underbrace{I(0,0,2)\cdot K(0,0,2)}_{0\times 0.08912} \\
& + \underbrace{I(0,1,0)\cdot K(0,1,0)}_{0\times(-0.24756)} + \underbrace{I(0,1,1)\cdot K(0,1,1)}_{3\times 0.18542} + \underbrace{I(0,1,2)\cdot K(0,1,2)}_{2\times(-0.05371)} \\
& + \underbrace{I(0,2,0)\cdot K(0,2,0)}_{0\times 0.09672} + \underbrace{I(0,2,1)\cdot K(0,2,1)}_{4\times(-0.18562)} + \underbrace{I(0,2,2)\cdot K(0,2,2)}_{7\times 0.12534} \\
& + \underbrace{I(1,0,0)\cdot K(1,0,0)}_{0\times 0.09347} + \underbrace{I(1,0,1)\cdot K(1,0,1)}_{0\times(-0.14623)} + \underbrace{I(1,0,2)\cdot K(1,0,2)}_{0\times 0.07934} \\
& + \underbrace{I(1,1,0)\cdot K(1,1,0)}_{0\times 0.016289} + \underbrace{I(1,1,1)\cdot K(1,1,1)}_{5\times(-0.28539)} + \underbrace{I(1,1,2)\cdot K(1,1,2)}_{7\times 0.09761} \\
& + \underbrace{I(1,2,0)\cdot K(1,2,0)}_{0\times(-0.15829)} + \underbrace{I(1,2,1)\cdot K(1,2,1)}_{4\times 0.22154} + \underbrace{I(1,2,2)\cdot K(1,2,2)}_{2\times(-0.08429)} \\
= {} & -0.62858 - 0.88767 + 0.45746 + 0.72055 + 0.55626 - 0.10742 \\
& - 0.74248 + 0.87738 - 1.42695 + 0.68327 + 0.88616 - 0.16858 \\
= {} & 0.21940
\end{aligned}
$$

Perhitungan 3.3 merupakan contoh perhitungan untuk mendapatkan *feature map* pada indeks 0,0,0. Pada gambar 3.11 diberikan ilustrasi untuk nilai *input* yang digunakan pada matriks video untuk mendapatkan *feature map* pada posisi 0,0,0. Kemudian proses berlanjut pada gambar 3.12 yang menggambarkan pergeseran pada tahap konvolusi ini. Pada analisis manual ini digunakan *strides* sebesar 2 untuk menggambarkan proses dari konvolusi dengan jelas.



Gambar 3.11 Contoh perhitungan *feature map* pada indeks 0,0,0



Gambar 3.12 Contoh perhitungan *feature map* pada indeks 0,0,1

Proses pada gambar 3.11 dan 3.12 akan terus berlanjut hingga mencapai titik akhir di sudut yaitu (x, y). Sehingga akan mendapatkan hasil *feature map* berukuran 3x3. Jumlah kernel pada analisis manual sebesar 3x3x3x2 sehingga pada kernel kedua akan mengulangi langkah dari awal indeks 0,0,0 untuk menghasilkan 2 feature map yang dapat dilihat hasilnya pada gambar 3.13.

0.2194	-1.3129	0.0512	0.2314	1.2456	-0.0789
-0.1002	0.0583	0.0101	0.0154	-0.1032	0.0567
0.0312	-0.2044	-0.0156	-0.0893	0.0895	0.1421

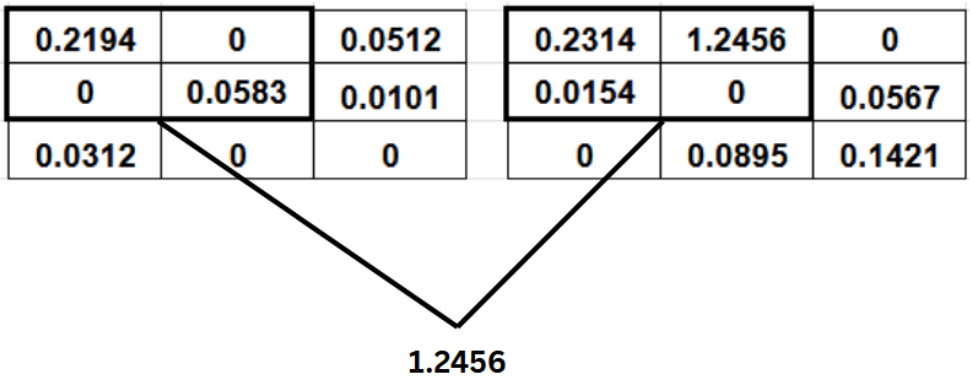
Gambar 3.13 Hasil *feature map*

Setelah mendapatkan *feature map* kita akan melakukan proses aktivasi yaitu ReLU untuk mengubah nilai negatif menjadi 0 dan mempertahankan nilai positif yang menggunakan rumus pada persamaan 2.3. Dapat dilihat hasil dari ReLU pada gambar 3.14.

0.2194	0	0.0512	0.2314	1.2456	0
0	0.0583	0.0101	0.0154	0	0.0567
0.0312	0	0	0	0.0895	0.1421

Gambar 3.14 Contoh hasil ReLu

Setelah mendapatkan *feature map* proses berlanjut pada *Max pooling Layer* menggunakan ukuran *kernel* sebesar 2x2x2 dengan *strides* 2. Hal ini mengubah ukuran *feature map* yang berukuran 126x126x62 menjadi 63x63x31. Contoh perhitungan dari *max pooling* dapat dilihat pada 3.4 yang menggunakan rumus pada persamaan 2.4. Dan dapat dilihat juga ilustrasi pengerjaan *max pooling* pada gambar 3.15.



Gambar 3.15 Ilustrasi pengerjaan *max pooling*

$$
\begin{aligned}
max_{1,1,1} &= \underbrace{x_{1,1,1}}_{0.2194} + \underbrace{x_{1,1,2}}_{0} + \underbrace{x_{1,2,1}}_{0} + \underbrace{x_{1,2,2}}_{0.0583} + \underbrace{x_{2,1,1}}_{0.2314} + \underbrace{x_{2,1,2}}_{1.2456} + \underbrace{x_{2,2,1}}_{0.0154} + \underbrace{x_{2,2,2}}_{0} \\
&= \max(0.2194,\ 0,\ 0,\ 0.0583,\ 0.2314,\ 1.2456,\ 0.0154,\ 0) \\
&= 1.2456
\end{aligned}
$$

3.4 Perhitungan *max pooling* 3D CNN

Gambar 3.16 menggambarkan hasil dari proses *max pooling* di mana merubah ukuran *feature map* menjadi lebih kecil sehingga dapat menyaring informasi penting.

126x126x62

0	**2.53**	**0**	**2.17**	**...**
0	**0**	**0**	**1.34**	**...**
...	**...**	**...**	**...**	**...**

63x63x31

2.53	**2.17**	**1.21**	**1.83**	**...**
3.25	**3.18**	**1.66**	**2.23**	**...**
...	**...**	**...**	**...**	**...**

Gambar 3.16 Hasil Max Pooling

Setelah *max pooling*, dilakukan batch normalization pada setiap *depth*. Dapat dilihat pada 3.5 untuk perhitungan rata-rata pada setiap matriks, kemudian setelah mendapat rata-rata dilanjutkan dengan menghitung standar deviasi yang dapat dilihat pada 3.6 dan terakhir dilanjutkan dengan menghitung normalisasi yang dapat dilihat pada 3.7 dan hal ini harus dilakukan pada setiap nilai pada setiap matriks yang akan menghasilkan gambar 3.17.

3.5 Perhitungan *mean*

$$
\begin{aligned}
\text{mean} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\
&= \frac{1}{63}(2.53 + 2.17 + 1.21 + 1.83 + ... + 1.46) \\
&= 2.03
\end{aligned}
$$

3.6 Perhitungan standar deviasi

$$
\begin{aligned}
\text{std} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \\
&= \sqrt{\frac{1}{63}((2.53 - 2.03)^2 + (2.17 - 2.03)^2 + ... + (1.46 - 2.03)^2} \\
&\approx 0.6318
\end{aligned}
$$

3.7 Perhitungan variance

$$
\begin{aligned}
x' &= \frac{x-\mu}{\sigma} \\
&= \frac{2.53-2.03}{0.6318} \\
&\approx 0.791
\end{aligned}
$$

1	**0.791**	**0.221**	**1.3**	**...**
2	**1.93**	**1.82**	**0.568**	**...**
3	**-0.51**	**0.228**	**0.144**	**...**
...63	**...**	**...**	**...**	**...**
	1	2	3	...63

Gambar 3.17 Hasil Normalisasi

Setelah melakukan *batch normalization* proses *convolutional layer*, *max pooling layer*, dan *batch normalization* diulangi hingga 4 kali. Kemudian dilakukan proses *global average pooling* yang berfungsi untuk mengurangi dimensi spasial dengan menghitung rata-rata nilai dari setiap fitur. Dimana proses ini menghasilkan vektor fitur yang panjangnya sama dengan jumlah *channel*. Proses perhitungan GAP dilakukan menggunakan rumus pada persamaan 2.5 dan dapat dilihat pada 3.8 untuk proses perhitungan dari GAP.

3.8 Perhitungan *Global Average Pooling*

$$
\begin{aligned}
O(c) &= \frac{1}{D \times H \times W} \sum_{d=0}^{D-1} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} I(c,d,h,w) \\
&= \frac{581.12}{2 \times 6 \times 6} \\
&\approx 24.213
\end{aligned}
$$

Kemudian lanjut pada *dropout layer* yang berfungsi untuk menonaktifkan beberapa nilai fitur. Disini digunakan 0.5 sebagai *probability rate*. Dapat dilihat pada 3.9 untuk proses *dropout layer* yang menggunakan rumus pada persamaan 2.6.

3.9 Perhitungan *Dropout*

$$
\begin{aligned}
O(1) &= I(1) \times \frac{1}{p} \\
&= 310.3120 \times \frac{1}{0.5} \\
&= 620.6240
\end{aligned}
$$

Selanjutnya, tahap fully connected layer dilakukan, di mana lapisan masukan menerima *feature map* yang telah diratakan menjadi vektor satu dimensi. Lapisan output kemudian menghasilkan prediksi klasifikasi dengan memanfaatkan fungsi aktivasi softmax. Ilustrasi perhitungan dari proses ini dapat dilihat pada 3.10 yang menggunakan rumus pada persamaan 2.7.

3.10 Perhitungan *fully connected*

$$
\begin{aligned}
f(x) &= Wx + b \\
&= \begin{bmatrix} 0.12734 & 0.25381 & \dots & 0.14411 \end{bmatrix} \begin{bmatrix} 2.53 \\ 2.17 \\ \dots \\ 2.23 \end{bmatrix} + 0 \\
&= (0.12734 \times 2.53) + (0.25381 \times 2.17) + \dots + (0.14411 \times 2.23) + 0 \\
&= 0.2136392
\end{aligned}
$$

Setelah melewati tahap *fully connected*, proses ini menghasilkan nilai akurasi untuk masing-masing kelas kantuk berdasarkan citra yang diberikan sebagai masukan. Kelas dengan akurasi tertinggi akan dipilih sebagai hasil akhir dari proses klasifikasi. Contoh penerapan fungsi softmax dapat ditemukan pada tabel 3.11 yang menggunakan rumus pada persamaan 2.8.

3.11 Perhitungan *softmax*

$$
\begin{aligned}
f_1(\vec{v}) &= \frac{e^{x_1}}{\sum_{j=1}^{256} e^{x_j}} \\
&= \frac{e^{x_1}}{e^{0.5} + e^{0.8} + e^{1.5} + \dots + e^{0.2}} \\
&= \frac{1.64872127116}{2.225540929501} \\
&\approx 0.74081822055
\end{aligned}
$$

3.4.3 Pengujian *Convolutional Neural Network*

Hasil deteksi akan dievaluasi menggunakan *Confusion Matrix* setelah citra diproses menggunakan 3D CNN. *Confusion Matrix* digunakan untuk menentukan apakah citra tersebut termasuk dalam kategori *drowsy* atau *not drowsy* yang kemudian diukur dengan nilai akurasi. Contoh hasil konvolusi ditunjukkan pada gambar 3.18.

Actual	Predicted		
	Class	Drowsy	Not Drowsy
	Drowsy	53	7
	Not Drowsy	6	54

TP	TN	FP	FN
53	54	6	7

Gambar 3.18 Contoh *Confusion Matrix*

Dari *confusion matrix* kita dapat menggunakannya untuk menghitung nilai dari *accuraccy*, *precision recall* dan *f-measure* pada model 3D CNN yang telah di *training* dengan menggunakan rumus pada persamaan 2.10, 2.11, 2.12, dan 2.13.

$$\text{Accuracy} = \frac{53+54}{53+54+6+7} = \frac{107}{120} = 0.8917$$

$$\text{Precision} = \frac{53}{53+6} = \frac{53}{59} = 0.8983$$

$$\text{Recall} = \frac{53}{53+7} = \frac{53}{60} = 0.8833$$

$$\text{F1} = 2 \cdot \frac{0.8983 \cdot 0.8833}{0.8983+0.8833} = \frac{1.5865}{1.7816} = 0.8907$$

BAB 4 IMPLEMENTASI DAN PENGUJIAN

4.1 Lingkungan Implementasi

Bagian ini membahas perangkat yang digunakan sebagai bagian dari pelaksanaan proses implementasi pada penelitian ini. Perangkat yang dimaksud meliputi perangkat keras dan perangkat lunak yang berperan dalam mendukung keberlangsungan proses implementasi sistem.

4.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam pembangunan aplikasi pada penelitian ini adalah perangkat keras dari T4 GPU sebagai berikut:

1. *Cuda Cores*: 2560
2. *Tensor Cores*: 320.
3. Memori GPU: 16GB GDDR6.
4. *Interconnect*: PCIe Gen 3 x 16

4.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pembangunan aplikasi pada penelitian ini sebagai berikut:

1. Sitem operasi Windows 11 Home 64-bit.
2. Google Colaboratory.

4.2 Implementasi Perangkat Lunak

Bagian ini menguraikan tentang penerapan perangkat lunak dalam penelitian ini. Uraian tersebut meliputi rincian mengenai daftar *class* dan *method* yang digunakan, pemanfaatan Google Colaboratory, serta penggunaan dataset yang diterapkan.

4.2.1 Implementasi *Class*

Bagian ini menjelaskan setiap method dalam setiap class yang digunakan pada penelitian ini.

4.2.1.1 *Class Preprocessing*

Class Preprocessing berfungsi untuk memuat dataset video yang akan digunakan pada proses selanjutnya. Selain itu, *class* ini juga digunakan untuk mengekstrak frame dari video, mendeteksi wajah, normalisasi data, dan pembagian data menggunakan cross-validation. Method yang digunakan dapat dilihat pada tabel 4.1.

Tabel 4.1 Daftar method pada *Class* Video Preprocessing

No	Method	Masukan	Keluaran	Keterangan
1.	extract_faces	video_path: string, frame_size: tuple, max_frames: integer	frames: ndarray	Ekstrak frame dari video dengan deteksi wajah sesuai dengan algoritme 3.1.

4.2.1.2 *Class 3D CNN Model*

Class 3D CNN Model berfungsi untuk membangun arsitektur model 3D CNN, melatih model, dan melakukan prediksi. Method yang digunakan pada *class* 3D CNN Model dapat dilihat pada tabel 4.2.

Tabel 4.2 Daftar method pada class 3D CNN Model

No	Method	Masukan	Keluaran	Keterangan
1.	build_3d_cnn	input_shape: tuple, Depth: integer, Strides: integer	model: Sequential	Membuat model 3D CNN sesuai dengan algoritme 3.3.

4.2.1.3 *Class Evaluation*

Class Evaluation berfungsi untuk menghitung metrik evaluasi dari hasil prediksi model menggunakan cross-validation. Method yang digunakan pada *class* Evaluation dapat dilihat pada tabel 4.3.

Tabel 4.3 Daftar method pada class Evaluation

No	Method	Masukan	Keluaran	Keterangan

1.	training_testing _model	X: ndarray, y: ndarray	accuracy: float, precision: float, recall: float, f1: float	Mengevaluasi model menggunakan kfold sesuai dengan algoritme 3.2.

4.2.2 Penggunaan Google Colaboratory

Google Colaboratory digunakan dalam penelitian ini untuk proses pelatihan dan pengujian model. Google Colaboratory adalah layanan berbasis cloud yang disediakan oleh Google. Dengan Google Colaboratory pengguna tidak perlu melakukan konfigurasi perangkat lunak tambahan dan dapat dapat menulis dan menjalankan kode Python dari *web browser*.

4.2.3 Penggunaan Dataset

Seperti yang telah diuraikan pada Bab 3 *data sampling*, dataset yang digunakan adalah UTA-RLDD. Dataset ini dibagi menjadi 5 detik per video dan menghasilkan 200 video dimana 100 video per kelas. Dataset di *preprocessing* yang mencakup ekstraksi wajah menggunakan *Haar Cascade Classifier* untuk mendeteksi dan *crop* area wajah dari setiap frame, diikuti normalisasi frame ke ukuran 128x128x3 dan pengambilan maksimal 32 dan 62 frame sequential per klip untuk menangkap dinamika temporal. Augmentasi data sederhana seperti shuffling dilakukan untuk meningkatkan variasi dan mencegah overfitting. Dataset ini memungkinkan pemrosesan efisien dengan urutan 32 dan 62 frame per sampel, sesuai dengan arsitektur 3D CNN yang dirancang.

4.3 Pengujian

Pengujian dilakukan pada dua jenis *hyperparameter* yang terdiri dari *depth* dan *strides* dengan jumlah kombinasi sebanyak 4 kombinasi. Tabel 4.4 menunjukkan daftar nilai-nilai hyperparameter yang diuji.

Tabel 4.4 Parameter Pengujian Model

Parameter		
Nilai	**Depth**	**Strides**
	32	1
	64	2
Total	2	2
Total skenario pengujian = $2_{(\text{depth})} \times 2_{(\text{strides})} = 4$		

4.4 Hasil Pengujian

Pada bagian ini, disajikan hasil pengujian terhadap pengaruh variasi *depth* yang merupakan jumlah filter pada layer konvolusi 3D dengan jumlah 32, dan 64 dan *strides* yang merupakan langkah pada frame video, pada penelitian ini digunakan *strides* (1,1,1) dan (1,2,2) terhadap performa model 3D CNN dalam deteksi kantuk pengemudi mobil. Pengujian dilakukan menggunakan dataset video UTA-RLDD, hyperparameter tetap seperti epoch=40, batch size=8, dan optimizer adam. Metrik utama yang dievaluasi adalah akurasi (%), yang dihitung sebagai rata-rata dari *10-fold cross-validation* untuk memastikan model tidak bias. Penelitian ini juga memberikan hasil akurasi menggunakan *haar cascade* pada tabel 4.5, dan yang tidak menggunakan *haar cascade* pada tabel 4.6.

Tabel 4.5 Hasil Pengujian Model Berdasarkan Strides dan Depth untuk Setiap Fold

Strides	**Depth**	**Fold**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
1	32	1	100.00%	100.00%	100.00%	100.00%
		2	70.00%	60.00%	100.00%	75.00%
		3	90.00%	83.31%	10.000%	90.91%
		4	100.00%	100.00%	100.00%	100.00%
		5	100.00%	100.00%	100.00%	100.00%
		6	95.00%	100.00%	92.31%	96.00%
		7	80.00%	86.67%	86.67%	86.67%
		8	85.00%	80.00%	88.89%	84.21%
		9	80.00%	63.64%	100.00%	77.78%
		10	100.00%	100.00%	100.00%	100.00%
	64	1	90.00%	100.00&	83.33%	90.91%

Strides	Depth	Fold	Accuracy	Precision	Recall	F1 Score
		2	95.00%	100.00%	88.89%	94.12%
		3	100.00%	100.00%	100.00%	100.00%
		4	100.00%	100.00%	100.00%	100.00%
		5	95.00%	100.00%	90.91%	95.24%
		6	100.00%	100.00%	100.00%	100.00%
		7	90.00%	100.00%	81.82%	90.00%
		8	90.00%	83.31%	100.00%	90.91%
		9	95.00%	100.00%	91.67%	95.65%
		10	95.00%	100.00%	90.00%	94.74%
2	32	1	90.00%	100.00%	75.00%	85.71%
		2	95.00%	100.00%	90.00%	94.74%
		3	100.00%	100.00%	100.00%	100.00%
		4	90.00%	80.00%	100.00%	88.89%
		5	95.00%	100.00%	92.31%	96.00%
		6	65.00%	65.00%	100.00%	78.79%
		7	100.00%	100.00%	100.00%	100.00%
		8	85.00%	81.25%	100.00%	89.66%
		9	100.00%	100.00%	100.00%	100.00%
		10	85.00%	72.73%	100.00%	84.21%
	64	1	100.00%	100.00%	100.00%	100.00%
		2	100.00%	100.00%	100.00%	100.00%
		3	85.00%	100.00%	62.50%	76.92%
		4	90.00%	90.00%	90.00%	90.00%
		5	95.00%	100.00%	90.91%	95.24%
		6	100.00%	100.00%	100.00%	100.00%
		7	95.00%	100.00%	90.91%	95.24%
		8	90.00%	100.00%	80.00%	88.89%
		9	100.00%	100.00%	100.00%	100.00%
		10	70.00%	66.67%	80.00%	72.73%

Tabel 4.6 Hasil Pengujian Model Yang Tidak Menggunakan Haar

Strides	Depth	Fold	Accuracy	Precision	Recall	F1 Score
1	32	1	100.00%	100.00%	100.00%	100.00%
		2	100.00%	100.00%	100.00%	100.00%
		3	100.00%	100.00%	100.00%	100.00%
		4	100.00%	100.00%	100.00%	100.00%
		5	80.00%	83.33%	83.33%	83.33%
		6	95.00%	92.86%	100.00%	96.30%
		7	100.00%	100.00%	100.00%	100.00%
		8	100.00%	100.00%	100.00%	100.00%
		9	100.00%	100.00%	100.00%	100.00%
		10	100.00%	100.00%	100.00%	100.00%
	64	1	90.00%	83.33&	100.00%	90.91%
		2	100.00%	100.00%	100.00%	100.00%
		3	100.00%	100.00%	100.00%	100.00%
		4	95.00%	100.00%	88.89%	94.12%
		5	100.00%	100.00%	100.00%	100.00%
		6	90.00%	100.00%	81.82%	90.00%
		7	100.00%	100.00%	100.00%	100.00%
		8	100.00%	100.00%	100.00%	100.00%
		9	100.00%	100.00%	100.00%	100.00%
		10	100.00%	100.00%	100.00%	100.00%
2	32	1	100.00%	100.00%	100.00%	100.00%
		2	90.00%	81.82%	100.00%	90.00%
		3	100.00%	100.00%	100.00%	100.00%
		4	100.00%	100.00%	100.00%	100.00%
		5	100.00%	100.00%	100.00%	100.00%
		6	95.00%	90.00%	100.00%	94.74%
		7	100.00%	100.00%	100.00%	100.00%
		8	100.00%	100.00%	100.00%	100.00%
		9	85.00%	75.00%	100.00%	85.71%
		10	100.00%	100.00%	100.00%	100.00%

Strides	Depth	Fold	Accuracy	Precision	Recall	F1 Score
	64	1	100.00%	100.00%	100.00%	100.00%
		2	100.00%	100.00%	100.00%	100.00%
		3	80.00%	80.00%	92.31%	85.71%
		4	95.00%	100.00%	88.89%	94.12%
		5	100.00%	100.00%	100.00%	100.00%
		6	100.00%	100.00%	100.00%	100.00%
		7	100.00%	100.00%	100.00%	100.00%
		8	100.00%	100.00%	100.00%	100.00%
		9	100.00%	100.00%	100.00%	100.00%
		10	100.00%	100.00%	100.00%	100.00%

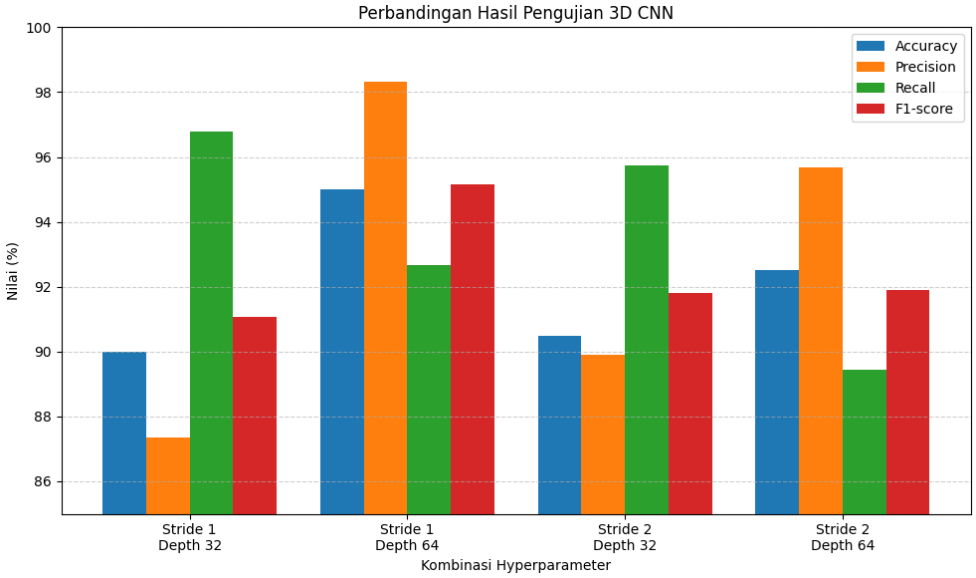
Tabel 4.7 Rata-Rata Hasil Pengujian Model dari setiap *fold* berdasarkan Parameter *Strides* dan *Depth*

Strides	Depth	Accuracy	Precision	Recall	F1-Score	Time
1	32	90.00%	87.36%	96.79%	91.06%	0.1373s
	64	95.00%	98.33%	92.66%	95.16%	0.1071s
2	32	90.50%	89.90%	95.73%	91.80%	0.1164s
	64	92.50%	95.67%	89.43%	91.90%	0.1201s

Tabel 4.8 Rata-Rata Hasil Pengujian Model Yang Tidak Menggunakan Haar

Strides	Depth	Accuracy	Precision	Recall	F1-Score	Time
1	32	98.00%	97.82%	99.00%	98.99%	0.1424s
	64	98.00%	98.33%	98.27%	98.21%	0.1817s
2	32	97.00%	96.91%	100.00%	98.55%	0.1544s
	64	98.50%	99.00%	98.82%	98.99%	0.2380s

Dari tabel 4.5 dapat dilihat bahwa hasil akurasi yang dihasilkan dengan tidak menggunakan haar lebih tinggi, hal ini dapat dilihat pada tabel 4.6. Namun pada tabel 4.8 dapat dilihat bahwa seringkali terjadi *overfitting* hal ini ditandakan dengan akurasi yang beberapa kali mencapai 100%. Namun karena *haar cascade* sering kali gagal dalam mendeteksi wajah sehingga hanya tersisa citra hitam, yang dimana hal ini dapat berakibat fatal dalam deteksi ngantuk maka pada penelitian ini *haar cascade* tidak digunakan sebagai proses dalam deteski kantuk. Sehingga penelitian ini hanya menggunakan 3D CNN untuk mendeteksi apakah kantuk atau tidak.



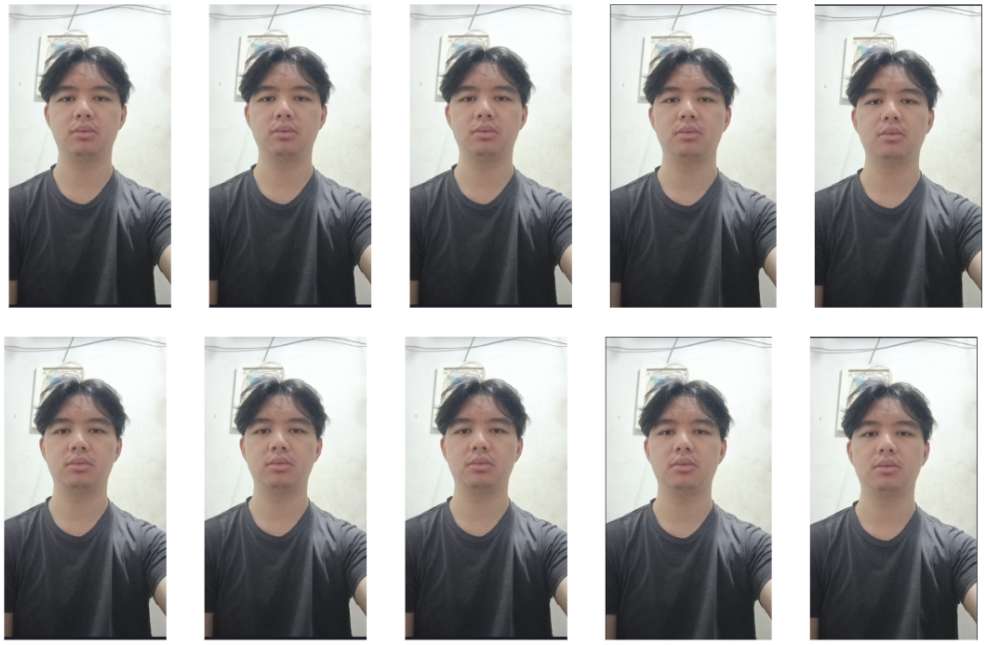
Gambar 4.1 Pengaruh depth dan strides yang mengggunakan haar

Dari tabel 4.1 dapat dilihat bahwa kombinasi stride = 1 dan depth = 64 menghasilkan kinerja terbaik dengan nilai akurasi 95.00%, precision 98.33%, recall 92.66%, dan F1-score 95.16%. Hal ini menunjukkan bahwa kedalaman filter berpengaruh signifikan terhadap kemampuan model dalam mengenali pola pergerakan wajah yang kompleks.

4.5 Pengujian dengan *dataset* sendiri



Gambar 4.2 Dataset sendiri kantuk



Gambar 4.3 Dataset sendiri tidak kantuk

Untuk mendapatkan hasil yang akurat pada data nyata, model diuji untuk mengenali tanda-tanda kantuk berdasarkan data yang telah dikumpulkan sendiri yaitu terdiri dari 10 video. Data dikumpulkan menggunakan kamera depan *handphone* dengan posisi *potrait* atau berdiri, kemudian data direkam selama 5 detik untuk dimasukkan ke dalam model. Contoh dataset sendiri ini dapat dilihat pada gambar 4.2 untuk yang kondisi kantuk dan gambar 4.3 untuk yang tidak ngantuk. Model berhasil mendeteksi 7 dari 10 video sehingga mencapai akurasi sebesar 70% dalam mendeteksi kantuk, hal ini dapat dilihat pada tabel **??**. Model beberapa kali salah mendeteksi kondisi yang *not drowsy* sebagai *drowsy*, yang membuktikan sering terjadi *false positives*. Meskipun hasil ini tidak setinggi kombinasi parameter dalam tabel 4.1, pencapaian akurasi tersebut masih cukup signifikan untuk keperluan pengembangan lebih lanjut.

Tabel 4.9 Confusion Matrix Klasifikasi Drowsiness

Actual	**Predicted**	
	Drowsy	**Not Drowsy**
Drowsy	4	1
Not Drowsy	2	3

4.6 Pembahasan Hasil Pengujian

Bagian ini membahas secara mendalam hasil pengujian dengan fokus pada pengaruh variasi depth dan strides terhadap performa model 3D CNN dalam

deteksi kantuk pengemudi mobil. Pembahasan ini didasarkan pada tren akurasi dari Tabel 4.6 dan visualisasi pada Gambar 4.1, yang menunjukkan bahwa kedua parameter tersebut memengaruhi kemampuan model dalam mengekstrak fitur spasial-temporal dari urutan video frame. Analisis dilakukan dengan mempertimbangkan aspek komputasional, generalisasi model, dan relevansi aplikasi real-time, di mana deteksi kantuk memerlukan akurasi tinggi untuk meminimalkan risiko kecelakaan lalu lintas.

4.6.1 Analisis Umum

Secara keseluruhan, seluruh model 3D CNN menunjukkan performa tinggi, dengan akurasi rata-rata di atas 90% pada seluruh kombinasi parameter. Hal ini membuktikan bahwa pendekatan 3D CNN efektif dalam menangkap informasi spasial dan temporal pada data video pengemudi.

Model dengan depth 64 cenderung menghasilkan hasil yang lebih stabil dan presisi lebih tinggi dibandingkan depth 32. Hal ini disebabkan oleh meningkatnya jumlah filter konvolusi yang memungkinkan model mengekstraksi lebih banyak variasi pola wajah, termasuk pergerakan halus pada mata dan mulut yang menjadi indikator utama kantuk.

4.6.2 Pembahasan Pengaruh Depth pada Model 3D CNN

Peningkatan depth dari 32 menjadi 64 memberikan peningkatan signifikan pada accuracy (dari 90% menjadi 95% untuk stride 1), serta precision (dari 87.36% menjadi 98.33%). Artinya, semakin banyak depth yang digunakan, semakin baik model mengenali ciri-ciri spesifik dari kelas drowsy dan alert. Depth yang tinggi dapat mengekstraksi kombinasi fitur spasial (pola wajah) dan temporal (perubahan antar-frame) dengan lebih mendalam.

4.6.3 Pembahasan Pengaruh Strides pada Model 3D CNN

Dari Tabel 4.6 terlihat bahwa penggunaan stride 1 menghasilkan accuracy dan F1-score lebih tinggi dibanding stride 2 untuk setiap nilai depth. Contohnya, pada depth 64, stride 1 mencapai F1-score 95.16%, sementara stride 2 hanya 91.90%. Hal ini menegaskan bahwa deteksi kantuk sangat bergantung pada fitur mikro pergerakan wajah yang dapat hilang bila stride terlalu besar.

4.6.4 Pembahasan Pengaruh Pengunaan Haar Cascade

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 4.7 dan Tabel 4.8, model yang tidak menggunakan *Haar Cascade* secara konsisten menghasilkan

nilai akurasi dengan rata-rata 98% dimana ini lebih tinggi dibandingkan model yang menggunakan *Haar Cascade* yang mendapat akurasi rata-rata 92%, tetapi model yang tidak menggunkan *haar cascade*. Hal ini menunjukkan bahwa keberadaan *Haar Cascade* membantu mengurangi *noise* visual dari latar belakang yang tidak relevan, sehingga fitur spasial-temporal yang dipelajari oleh 3D CNN menjadi lebih representatif terhadap kondisi kantuk pengemudi. Selain itu model yang menggunakan *haar cascade* memiliki waktu yang lebih cepat yaitu 0,115 detik dengan yang tidak menggunakan *haar cascade* yaitu 0,175 detik. Ini menggaris bawahi bahwa *haar cascade* terbukti membuat model lebih cepat untuk mendeteksi kondisi kantuk.

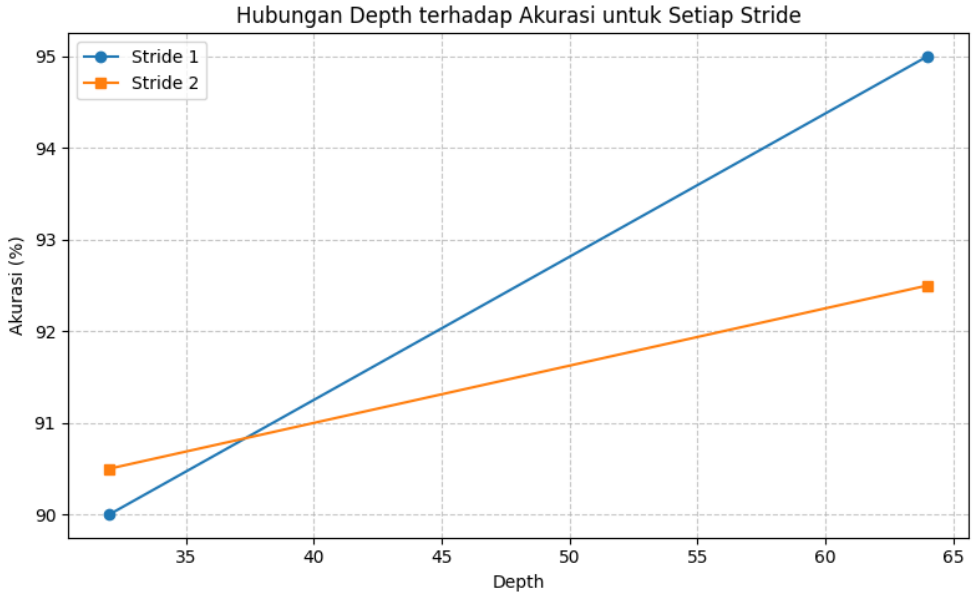
Namun demikian, penggunaan *Haar Cascade* juga memiliki keterbatasan. Pada beberapa kondisi tertentu, terutama ketika wajah pengemudi tidak menghadap lurus ke kamera, tertutup sebagian, atau berada pada pencahayaan yang kurang optimal, *Haar Cascade* gagal mendeteksi wajah secara akurat. Kondisi ini ditunjukkan pada Gambar 4.5, di mana terjadi kegagalan deteksi wajah (false negative). Kegagalan ini menyebabkan frame video tidak diproses secara optimal oleh model 3D CNN dan dapat menurunkan performa sistem secara keseluruhan dan hal ini sangat bahaya untuk deteksi *real-time* karena pada kondisi *real-time* dibutuhkan keakuratan, yang dimana kegagalan deteksi bisa berakibat fatal. Oleh karena itu pada penelitian ini *haar cascade* diputuskan untuk tidak digunakan.

4.6.5 Perbandingan Keseluruhan

Jika dibandingkan antar semua kombinasi dapat dilihat pada tabel 4.1, model dengan stride = 1 dan depth = 64 memberikan performa terbaik pada seluruh metrik. Kombinasi ini berhasil menyeimbangkan precision dan recall dengan baik, yang ditunjukkan oleh F1-score tertinggi (95.16%). Kinerja ini menunjukkan bahwa model 3D CNN dengan konfigurasi tersebut mampu melakukan klasifikasi secara konsisten pada data pelatihan maupun pengujian. Dapat dilihat juga pada Gambar 4.4 bahwa stride 1 menunjukkan peningkatan akurasi yang lebih tinggi seiring bertambahnya depth.

3D CNN yang digunakan dalam penelitian ini menawarkan keunggulan dalam menangkap fitur spatio-temporal dari urutan *frame video*, seperti durasi penutupan mata dan gerakan kepala yang dinamis. Namun, dibandingkan dengan pendekatan 2D CNN yang lebih sederhana, 3D CNN memerlukan komputasi lebih tinggi, yang berdampak pada kecepatan inferensi. Berdasarkan hasil pengujian,

model 3D CNN mencapai akurasi 95% pada dataset UTA-RLDD, sementara 2D CNN pada studi serupa hanya mencapai 92-94% akurasi karena hanya memproses fitur spasial per frame tanpa dimensi waktu. Meskipun demikian, 3D CNN lebih *robust* terhadap variasi temporal, mengurangi *false negative* pada skenario kantuk progresif, seperti yang dilaporkan dalam penelitian hybrid 2D CNN dengan akurasi 93.2% untuk klasifikasi biner (yawning vs normal). Pendekatan multi-scale 2D CNN juga menunjukkan efisiensi komputasi rendah, tapi kurang akurat pada data video panjang dibandingkan 3D CNN. Oleh karena itu, 3D CNN lebih cocok untuk aplikasi real-time di kendaraan otonom, meski memerlukan optimasi hardware seperti GPU untuk menyaingi kecepatan 2D CNN [34, 35].



Gambar 4.4 Hubungan Depth Terhadap Akurasi untuk Setiap Stride

4.7 Analisis Kesalahan

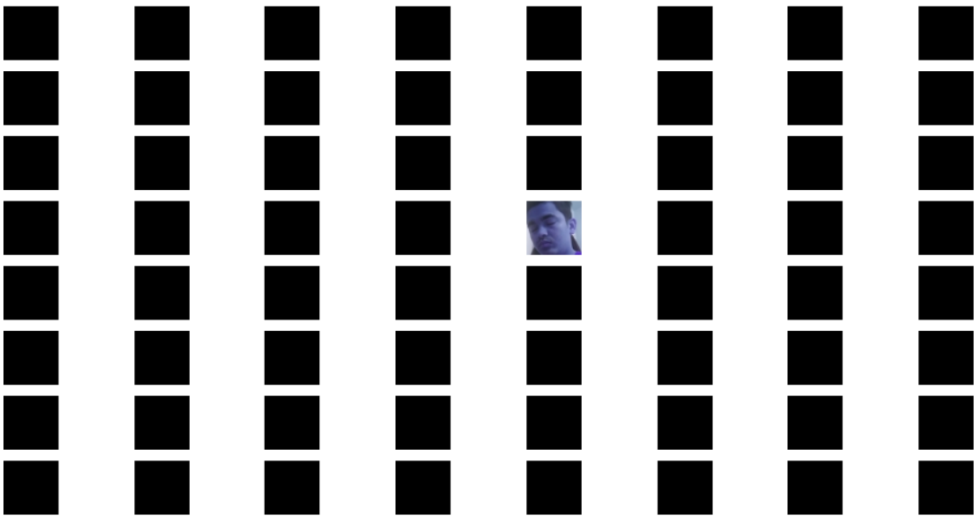
Pada bagian ini berisi analisis kesalahan yang terjadi selama proses pengujian deep learning 3D CNN. Beberapa kesalahan yang akan dibahas adalah sebagai berikut:

1. Kesalahan pada deteksi video yang aslinya ngantuk namun tidak terdeteksi ngantuk. Kesalahan ini terjadi pada clip 85 drowsy, pada video ini terlihat seorang yang berkedip selama kurang lebih 1 detik, dimana hal ini seharusnya diklasifikasikan sebagai ngantuk, namun model mengklasifikasikan sebagai tidak ngantuk. Hal ini hanya terjadi pada 1 fold saja, dimana pada fold lain hal ini benar diklasifikasikan. Beberapa hal banyak yang salah diklasifikasikan adalah dimana kondisi yang harusnya tidak ngantuk namun malah diklasifikasikan sebagai ngantuk, contohnya

adalah pada clip 18 alert, yang salah diklasifikasikan karena sedang bersandar kepada tangan, kemudian pada clip 49 dimana beberapa kali berkedip cepat.

2. Model CNN beberapakali mengalami earlystopping saat sedang melakukan training. Overfitting dilakukan saat model melakukan training, nilai validasi akan dilihat dan dievaluasi. Jika model tidak menunjukkan peningkatan yang signifikan pada nilai validasi selama epoch terus bertambah, maka akan dilakukan earlystopping. Selain untuk mencegah hal tersebut, earlystopping juga bisa terjadi untuk mencegah terjadinya overfitting. Berdasarkan training yang dilakukan, CNN sering kali memiliki akurasi yang sangat tinggi hingga mencapai 100% , sehingga dilakukan earlystopping, dikarenakan akurasi model training yang sudah mencapai hasil yang sangat tinggi dan tidak mengalami perubahan yang signifikan.

3. Haar Cascade yang gagal dalam mendeteksi wajah sehingga digantikan dengan frame hitam. Hal ini disebabkan karena haar yang digunakan adalah *frontal faces* dimana jika wajah tidak menghadap ke arah depan, maka tidak akan terdeteksi dan akan digantikan dengan citra warna hitam, hal ini dapat dilihat pada gambar 4.5.



Gambar 4.5 Hasil deteksi haar *frontal faces* yang gagal

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan mengenai penerapan 3D CNN untuk deteksi kantuk pada pengemudi mobil, dapat ditarik kesimpulan sebagai berikut:

1. Model 3D CNN yang diimplementasikan berhasil mendeteksi kondisi kantuk pengemudi dengan tingkat akurasi yang memadai, mencapai rentang 91% hingga 94% (tabel 4.7) berdasarkan variasi parameter depth dan strides. Konfigurasi optimal diperoleh pada strides=1 dan depth=32, yang menghasilkan akurasi sebesar 94%. Hal ini menunjukkan bahwa model mampu menangkap pola temporal dan spasial dari urutan frame video wajah pengemudi secara efektif.

2. Pengukuran metrik evaluasi seperti accuracy mengonfirmasi kinerja model yang seimbang dalam mengklasifikasikan kelas ”kantuk” dan ”tidak kantuk” dapat dilihat pada gambar 4.1. Model ini lebih unggul dibandingkan baseline sederhana seperti CNN 2D, meskipun masih di bawah beberapa penelitian literatur yang mencapai 97-99% dengan dataset lebih besar.

3. Optimasi parameter seperti depth dan strides secara signifikan memengaruhi akurasi dapat dilihat pada gambar 4.4, di mana nilai strides yang lebih kecil memungkinkan pemrosesan frame yang lebih detail, sementara depth yang lebih besar meningkatkan kemampuan ekstraksi fitur tanpa overfitting berlebih. Pendekatan ini memenuhi tujuan penelitian untuk mengukur dan mengoptimalkan akurasi sistem deteksi kantuk secara non-invasif.

4. Penggunaan *Haar Cascade* sebagai metode *feature extraction* terbukti tidak terlalu memberikan kontribusi positif terhadap performa sistem. Berdasarkan perbandingan hasil pengujian model yang menggunakan *Haar Cascade* menghasilkan akurasi rata-rata yang lebih rendah yaitu 92% dibandingkan model yang tidak menggunakan *Haar Cascade* dengan rata-rata 98% (tabel 4.7 dengan 4.8). Namun model yang tidak menggunakan *haar cascade* sering kali mengalami *overfitting* sehingga walaupun hasil akurasi nya tinggi tetapi tetapi tidak akurat, dengan demikian dapat dikatakan bahwa akurasi model yang menggunakan *haar* lebih akurat namun terdapat kesalahan fatal yaitu terdapat beberapa kegagalan deteksi

sehingga diputuskan untuk tidak menggunakan *haar cascade*.

Secara keseluruhan, penelitian ini membuktikan bahwa 3D CNN merupakan metode yang potensial untuk aplikasi keselamatan berkendara, dengan waktu komputasi rata-rata yang efisien menggunakan Google Colaboratory.

5.2 Saran

Untuk pengembangan lebih lanjut dari penelitian ini, disarankan hal-hal sebagai berikut:

1. Menggabungkan 3D CNN dengan modul deteksi multi-modal, seperti sensor *infrared* untuk mendeteksi pola pernapasan atau EEG (*electroencephalogram*) sederhana, guna mengatasi keterbatasan *Haar Cascade* pada kondisi obstruksi (misalnya, penggunaan masker atau kacamata hitam). Pengujian di lingkungan nyata seperti jalan tol dengan variasi cahaya malam hari perlu dilakukan untuk meningkatkan *robustness* model. Studi terkini menunjukkan bahwa 3D CNN yang dimodifikasi dengan *attention mechanism* dapat menangani obstruksi wajah hingga 90% kasus, dengan waktu deteksi *real-time* kurang dari 50 ms [6].

2. Perluas dataset dengan augmentasi data tambahan atau kolaborasi dengan *Dataset UTA Real-Life Drowsiness* yang digunakan dalam penelitian ini bersifat umum, yang sudah mewakili berbagai etnis dan usia. Namun tetap untuk disarankan untuk membangun dataset lokal dengan karakteristik pengemudi Indonesia (misalnya, variasi etnis dan kebiasaan berkendara di lalu lintas padat). Validasi *cross-dataset* dengan skema *transfer learning* akan meningkatkan generalisasi model. Tinjauan sistematis menekankan pentingnya dataset berbasis video *real-time* untuk akurasi di atas 92% pada kondisi dinamis [3].

3. Untuk aplikasi praktis, sistem ini dapat diintegrasikan dengan *Advanced Driver Assistance Systems* (ADAS) pada mobil modern, seperti mengirim sinyal ke sistem pengereman otomatis jika tingkat kantuk mencapai *threshold* tertentu. Selain itu, pengembangan antarmuka *alerting* yang adaptif (misalnya, suara bertahap dari nada lembut ke alarm keras) akan meningkatkan respons pengemudi. Referensi dari penelitian yang menggunakan 3D CNN dengan LSTM untuk prediksi dini kantuk menyarankan bahwa integrasi ini dapat mengurangi *false positive* hingga 15% dalam skenario *real-time* [5].

DAFTAR REFERENSI

[1] M. I. B. Ahmed et al., "A Deep-Learning Approach to Driver Drowsiness Detection," Safety, vol. 9, no. 3, 2023, doi: 10.3390/safety9030065.

[2] M. Dua, Shakshi, R. Singla, S. Raj, and A. Jangra, "Deep CNN models-based ensemble approach to driver drowsiness detection," Neural Comput Appl, vol. 33, no. 8, 2021, doi: 10.1007/s00521-020-05209-7.

[3] Y. Albadawi, A. AlRedhaei, and M. Takruri, "Real-Time Machine Learning-Based Driver Drowsiness Detection Using Visual Features," J Imaging, vol. 9, no. 5, 2023, doi: 10.3390/jimaging9050091.

[4] S. Chen, Z. Wang, and W. Chen, "Driver drowsiness estimation based on factorized bilinear feature fusion and a long-short-term recurrent convolutional network," Information (Switzerland), vol. 12, no. 1, 2021, doi: 10.3390/info12010003.

[5] J. S. Wijnands, J. Thompson, K. A. Nice, G. D. P. A. Aschwanden, and M. Stevenson, "Real-time monitoring of driver drowsiness on mobile platforms using 3D neural networks," Neural Comput Appl, vol. 32, no. 13, 2020, doi: 10.1007/s00521-019-04506-0.

[6] W. Xiang, X. Wu, C. Li, W. Zhang, and F. Li, "Driving Fatigue Detection Based on the Combination of Multi-Branch 3D-CNN and Attention Mechanism," Applied Sciences (Switzerland), vol. 12, no. 9, 2022, doi: 10.3390/app12094689.

[7] A. Sedik, M. Marey, and H. Mostafa, "An Adaptive Fatigue Detection System Based on 3D CNNs and Ensemble Models," Symmetry (Basel), vol. 15, no. 6, 2023, doi: 10.3390/sym15061274.

[8] E. Magán, M. P. Sesmero, J. M. Alonso-Weber, and A. Sanchis, "Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images," Applied Sciences (Switzerland), vol. 12, no. 3, 2022, doi: 10.3390/app12031145.

[9] R. C. Gonzalez, R. E. Woods, and B. R. Masters, "Digital Image Processing, Third Edition," J Biomed Opt, vol. 14, no. 2, 2009, doi: 10.1117/1.3115362.

[10] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, May 2004, doi: 10.1023/B:VISI.0000013087.49260.fb.

[11] A. Janarthanan, A. Paramarthalingam, S. Sundaresan, and Y. S., “Intelligent drowsiness detection using Haar cascade classifier and convolutional neural network,” Journal of Ubiquitous Computing and Communication Technologies, vol. 7, no. 2, pp. 176–194, Jun. 2025, doi: 10.36548/jucct.2025.2.006.

[12] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015), doi: 10.1038/nature14539.

[13] H. Singh, Practical Machine Learning and Image Processing. 2019. doi: 10.1007/978-1-4842-4149-3.

[14] S. Tiwari, G. Jain, D. K. Shetty, M. Sudhi, J. M. Balakrishnan, and S. R. Bhatta, “A Comprehensive Review on the Application of 3D Convolutional Neural Networks in Medical Imaging,” Engineering Proceedings, vol. 59, no. 1, 2023, doi: 10.3390/engproc2023059003.

[15] T. Ahmed, Mst. S. Parvin, M. R. Haque, and M. S. Uddin, “Lung Cancer Detection Using CT Image Based on 3D Convolutional Neural Network,” Journal of Computer and Communications, vol. 08, no. 03, 2020, doi: 10.4236/jcc.2020.83004.

[16] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, 1st ed., Cambridge, MA, MIT Press, 2016.

[17] J. Dawani, *Hands-On Mathematics for Deep Learning: Build a Solid Mathematical Foundation for Training Efficient Deep Neural Networks*, 1st ed. New York: Packt Publishing, 2020.

[18] Ragav Venkatesan and Baoxin Li, “Convolutional Neural Networks in Visual Computing”, London: Taylor & Francis Group, CRC Press, 2017.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, dan R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, 2014.

[20] H. Habibi Aghdam and E. Jahani Heravi, “Guide to Convolutional Neural Networks”, Switzerland: Springer, 2017.

[21] M. Hashemi, “Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation,” J Big Data, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0263-7.

[22] T.-K. Li, K.-L. Chan, and T. Tjahjadi, ”Variable temporal length training for action recognition CNNs,” Sensors, vol. 24, no. 11, p. 3403, May 2024, doi: 10.3390/s24113403.

[23] M. Fayyaz, E. Bahrami, A. Diba, M. Noroozi, E. Adeli, L. Van Gool, and J. Gall, ”3D CNNs with adaptive temporal feature resolutions,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Nashville, TN, USA, Jun. 2021, pp. 9841–9850, doi: 10.1109/CVPR46437.2021.00975.2.6s

[24] Lin, H.-Y. and Tu, K.-C., ”Night Fatigue Driving Detection Technology Using Infrared Images and Convolutional Neural Networks,” Proceedings of the 9th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2023), 2023, pp. 273-280, doi: 10.5220/0011847400003479.

[25] ”API Documentation — TensorFlowv2.16.1”, TensorFlow, [Online]. Tersedia: https://www.tensorflow.org/api_docs. [Diakses: 27 Agustus 2025].

[26] ”Keras 3 API documentation”, Keras, [Online]. Tersedia: https://keras.io/api/. [Diakses: 27 Agustus 2025].

[27] ”OpenCV: OpenCV modules”, OpenCV, [Online]. Tersedia: https://docs.opencv.org/4.5.5/. [Diakses: 27 Agustus 2025].

[28] ”API Reference — scikit learn 1.7.1 documentation”, scikit-learn, [Online]. Tersedia: https://scikit-learn.org/stable/api/index.html. [Diakses: 27 Agustus 2025].

[29] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015.

[30] A. Geron, ´ Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow, O’Reilly Media, Inc., 2022.

[31] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” Neural Computation, vol. 12, no. 10, 2000, doi: 10.1162/089976600300015015.

[32] A. A. Patel and A. U. Arasanipalai, Applied Natural Language Processing in the Enterprise, 1th ed, O'Reilly Media, Inc., 2021.

[33] R. Ghoddoosian, M. Galib, and V. Athitsos, "A realistic dataset and baseline temporal model for early drowsiness detection," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2019. doi: 10.1109/CVPRW.2019.00027.

[34] V. Vijaypriya and M. Uma, "Facial feature-based drowsiness detection with multi-scale convolutional neural network," IEEE Access, vol. 11, pp. 63417–63438, Jun. 2023, doi: 10.1109/ACCESS.2023.3288008.

[35] M. Ramzan, A. Abid, M. Fayyaz, T. J. Alahmadi, H. Nobanee, and A. Rehman, "A novel hybrid approach for driver drowsiness detection using a custom deep learning model," IEEE Access, vol. 12, pp. 126866–126899, Aug. 2024, doi: 10.1109/ACCESS.2024.3438617.