

**IMPLEMENTASI ALGORITMA DEEP LEARNING UNTUK
SISTEM DETEKSI KANTUK PADA PENGEMUDI MENGGUNAKAN YOLO**

TUGAS AKHIR

**Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Strata – 1
Pada Program Studi Teknik Industri Fakultas Teknologi Industri**



Disusun oleh:

Nama : Mamta Anisa Bella

NIM : 17522216

**JURUSAN TEKNIK INDUSTRI
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA**

2021

SURAT KETERANGAN PENELITIAN



FAKULTAS
TEKNIK INDUSTRI

Gedung KH. Mas Mansur
Jl. Kaliurang Km 14,5 Yogyakarta
Telp. (0274) 895287, 898444 ext 2511;
Fax. (0274) 895007

SURAT KETERANGAN PENELITIAN

Nomor : 226/A/Ka.Lab DATMIN/FTI-UII/XII/2021

Assalamu'alaikum Warahmatullahi Wabarakatuh

Kami yang bertanda tangan dibawah ini, menerangkan bahwa mahasiswa dengan keterangan sebagai berikut :

Nama : Mamta Anisa Bella
No. Mhs : 17522216
Dosen Pembimbing : Andrie Pasca Hendradewa, S.T., M.T

Telah selesai melaksanakan penelitian yang berjudul " Implementasi Algoritma Deep Learning untuk Sistem Deteksi Kantuk pada Pengemudi Menggunakan YOLO" di Laboratorium Data Mining, Program Studi Teknik Industri, Fakultas Teknologi Industri, Universitas Islam Indonesia tercatat mulai tanggal 01 Oktober 2021 sampai dengan tanggal 31 Oktober 2021.

Demikian surat keterangan kami keluarkan, agar dapat dipergunakan sebagaimana mestinya.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 03 Jumadil-Ula 1443 H
08 Desember 2021 M

Kepala Laboratorium
Data Mining

Annisa Uswatun Khasanah, S.T., M.Sc.

PERNYATAAN KEASLIAN

Demi Allah, saya akui karya ini adalah hasil kerja saya sendiri kecuali nukilan dan ringkasan yang setiap satunya telah saya jelaskan sumbernya. Jika dikemudian hari ternyata terbukti pengakuan saya ini tidak benar dan melanggar peraturan yang sah dalam karya tulis dan hak kekayaan intelektual maka saya bersedia ijazah yang telah saya terima untuk ditarik kembali oleh Universitas Islam Indonesia.

Yogyakarta, Desember 2021



Mamta Anisa Bella

LEMBAR PENGESAHAN PEMBIMBING

**Implementasi Algoritma Deep Learning untuk Sistem Deteksi Kantuk pada
Pengemudi Menggunakan YOLO**

TUGAS AKHIR



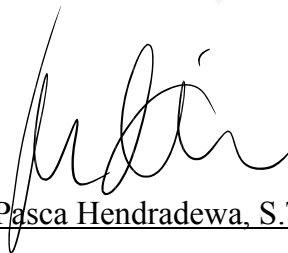
Disusun Oleh:

Nama : Mamta Anisa Bella

NIM : 17522216

Yogyakarta, Desember 2021

Dosen Pembimbing



Andrie Pasca Hendradewa, S.T., M.T.

LEMBAR PENGESAHAN PENGUJI**Implementasi Algoritma Deep Learning untuk Sistem Deteksi Kantuk pada
Pengemudi Menggunakan YOLO****TUGAS AKHIR**

Oleh

Nama : Mamta Anisa Bella

NIM : 17522216

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk
memperoleh gelar Sarjana Strata-1 Teknik Industri

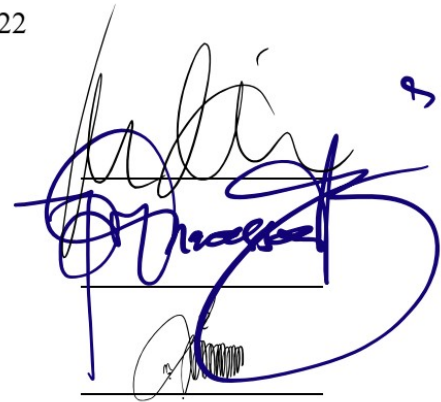
Yogyakarta, Januari 2022

Tim Penguji

Andrie Pasca Hendradewa, S.T., MT
Ketua

Ir. Ira Promasanti RD, M.Eng
Anggota I

Abdullah 'Azzam, S.T., M.T.
Anggota II



Mengetahui

Ketua Program Studi Teknik Industri
Fakultas Teknologi Industri
Universitas Islam Indonesia



Immawan, S.T., M.M

HALAMAN PERSEMBAHAN

Mengucapkan terimakasih kepada pemilik alam semesta, Allah SWT, atas izin dan kehendaknya-Nya saya persembahkan tugas akhir ini kepada kedua orang tua saya yang selalu kebersamai melalui untaian doa, kehadiran dan kasih sayang dengan macam rupa. Teruntuk kakak, adik, dan tidak lupa kemenakan saya yang selalu menjadi obat lelah ketika rasa pasrah dan ingin menyerah sesekali menghampiri. Saya persembahkan juga juga tulisan ini untuk sahabat dan teman yang sudah memberi warna pada kanvas cerita saya.



HALAMAN MOTTO

All we have is now



KATA PENGANTAR

Assalamu'alaikum Warakhmatullahi Wabarakatuh

Alhamdulillah, segala puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat, taufiq dan hidayah-Nya, sehingga penulis dapat menyusun dan menyelesaikan Tugas Akhir yang berjudul **“Implementasi Algoritma Deep Learning untuk Sistem Deteksi Kantuk pada Pengemudi Menggunakan YOLO”**.

Tugas akhir ini dilakukan sebagai salah satu persyaratan untuk menyelesaikan jenjang Strata-1 di Jurusan Teknik Industri Universitas Islam Indonesia. Dalam pelaksanaan dan penyusunan laporan tugas akhir, penulis banyak mendapatkan bantuan dan dukungan dari berbagai pihak. Untuk itu penulis ingin mengucapkan banyak terima kasih kepada:

1. Bapak Prof. Dr. Ir. Hari Purnomo, M.T. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph. D. selaku Ketua Jurusan Teknik Industri Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Dr. Taufiq Immawan, S.T., M.M. selaku Ketua Program Studi Sarjana Teknik Industri Fakultas Teknologi Industri, Universitas Islam Indonesia.
4. Bapak Andrie Pasca Hendradewa S.T., M.T. selaku dosen pembimbing tugas akhir yang telah memberikan ilmu, bimbingan, saran, serta waktunya dalam penyusunan Tugas Akhir.
5. Bapak dan Ibu, terima kasih untuk segala bentuk doa, materi, dan kasih sayang yang membawa penulis berada pada bab kehidupan saat ini.
6. Sahabat, teman dan kerabat yang telah membantu dalam penyelesaian tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih terdapat banyak kekurangan oleh karenanya kritik dan saran yang membangun sangat diharapkan. Semoga laporan ini dapat bermanfaat bagi pembaca. *Aamiin*

Wassalamu'alaikum Warrahmaatullahi Wabarakatuh.

Yogyakarta, Desember 2021



Mamta Anisa Bella

ABSTRAK

Mengemudi dalam keadaan mengantuk merupakan salah satu tindakan berbahaya yang dapat menjadi salah satu penyebab terjadinya kecelakaan yang dapat mengakibatkan cedera hingga meregang nyawa. Pencegahan kecelakaan yang diakibatkan kantuk pada pengemudi dapat dilakukan dengan cara melakukan deteksi kantuk sejak dini secara akurat. Seiring berkembangnya teknologi, dalam beberapa dekade terakhir kecerdasan buatan untuk melakukan deteksi kantuk pada pengemudi menjadi subjek banyak penelitian dan berbagai metode telah dikembangkan. Mengimplementasikan arsitektur *convolutional neural network* (CNN) dengan algoritma *you only look once* (YOLO) menjadi salah satu langkah yang dapat digunakan untuk melakukan analisis deteksi kantuk pada pengemudi yang dapat diterapkan melalui cuplikan video maupun secara realtime. Pada penelitian ini kategori kelas deteksi dibagi menjadi dua yaitu pengemudi yang terdeteksi mengantuk akan diberi label *drowsy* sedangkan pengemudi yang terdeteksi terjaga atau dalam kondisi normal akan diberi label *awake* sesuai dengan *bounding box* yang sudah dibuat sebelumnya. Selain itu dengan melakukan konfigurasi parameter model dengan *batch size* 64, *network size* 416x416, *subdivisions* 16, *max batch* 4000, *filters* 21, dengan empat skenario training data dan testing data dengan learning rate 0.00261 dan 0.001 dengan split dataset 90%:10% dan 80%:10% diperoleh nilai IoU terbesar pada konfigurasi menggunakan split dataset 80%:20% dengan learning rate 0.00261.

Kata kunci : *deep learning*, *cnn*, *yolo*, deteksi objek, deteksi kantuk

DAFTAR ISI

SURAT KETERANGAN PENELITIAN.....	II
PERNYATAAN KEASLIAN	III
LEMBAR PENGESAHAN PEMBIMBING.....	IV
LEMBAR PENGESAHAN PENGUJI.....	V
HALAMAN PERSEMBAHAN	VI
HALAMAN MOTTO.....	VII
KATA PENGANTAR.....	VIII
ABSTRAK.....	IX
DAFTAR ISI.....	X
DAFTAR TABEL.....	XIII
DAFTAR GAMBAR.....	XIV
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah.....	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	4
1.6 Sistematika Penulisan.....	4
BAB II KAJIAN LITERATUR.....	6
2.1 Kajian Deduktif.....	6
2.1.1 <i>Artificial Intelligence</i>	6
2.1.2 <i>Machine Learning</i>	6
2.1.3 <i>Deep Learning</i>	7
2.1.4 <i>Computer Vision</i>	7
2.1.5 <i>Object Detection</i>	7
2.1.6 <i>Convolutional Neural Network (CNN)</i>	8
2.1.7 <i>Convolutional Layers</i>	9
2.1.8 <i>Fully Connected Layers</i>	10
2.1.9 <i>Pooling Layers</i>	11

2.1.10	<i>You Only Look Once (YOLO)</i>	12
2.1.11	<i>Google Colaboratory</i>	14
2.2	Kajian Induktif	15
BAB III METODE PENELITIAN		18
3.1	Objek Penelitian	18
3.2	Metode Pengumpulan Data	18
3.3	Sumber Data	18
3.4	Alur Penelitian	18
3.4.1	Identifikasi Masalah	21
3.4.2	Studi Literatur	21
3.4.3	Pengumpulan Data	21
3.4.4	Pengolahan Data	21
3.4.5	Analisis Data	21
3.4.6	Kesimpulan dan Saran	22
BAB IV PENGUMPULAN DAN PENGOLAHAN DATA		23
4.1	Pengumpulan Dataset	23
4.2	<i>Pre-processing</i> Data	23
4.2.1	Pelabelan Dataset	24
4.2.2	<i>Split</i> Dataset	26
4.3	Konfigurasi <i>Google Colaboratory</i>	27
4.4	Konfigurasi Parameter Model	27
4.4.1	<i>Batch size</i>	28
4.4.2	<i>Network size</i>	28
4.4.3	<i>Subdivisions</i>	28
4.4.4	<i>Max batch</i>	29
4.4.5	<i>Filters</i>	29
4.4.6	<i>Class</i>	30
4.4.7	<i>Learning Rate</i>	30
4.5	<i>Processing Data</i>	30
4.5.1	<i>Training Dataset</i>	30
4.5.2	<i>Testing Dataset</i>	31
BAB V ANALISIS DAN PEMBAHASAN		38

BAB VI PENUTUP	41
6.1 Kesimpulan	41
6.2 Saran.....	41
DAFTAR PUSTAKA	42
LAMPIRAN.....	44



DAFTAR TABEL

Tabel 2. 1. Tabel kajian induktif.....	17
Tabel 4. 1. Konfigurasi parameter model	27
Tabel 4. 2. Tabel skenario training dataset	31



DAFTAR GAMBAR

Gambar 2. 1. Contoh deteksi pada objek	8
Gambar 2. 2. Arsitektur CNN untuk computer vision	9
Gambar 2. 3. Convolution layer dengan satu buah filter berukuran 5 x 5	10
Gambar 2. 4. Convolution layer dengan empat buah filter berukuran 5 x 5	10
Gambar 2. 5. Lapisan fully connected layer	11
Gambar 2. 6. Proses pooling untuk mereduksi dimensi data	11
Gambar 2. 7. Teknik max pooling dan average pooling	12
Gambar 2. 8. Intersection over union	13
Gambar 2. 9. Deteksi objek pada sistem menggunakan YOLO	14
Gambar 2. 10. Proses deteksi objek menggunakan YOLO	14
Gambar 2. 11. Arsitektur jaringan pada YOLOv4	14
Gambar 3. 1. Diagram Alir Penelitian	19
Gambar 3. 2 Diagram Alir Penelitian Lanjutan	20
Gambar 4. 1. Gambar wajah mengantuk (drowsy) dan terjaga (awake)	23
Gambar 4. 2. Diagram proses pelabelan dataset	24
Gambar 4. 3. Proses input gambar	25
Gambar 4. 4. Pembuatan bounding box untuk kelas mengantuk (drowsy)	25
Gambar 4. 5. Pembuatan bounding box untuk kelas terjaga (awake)	26
Gambar 4. 6. Testing pada gambar 1 skenario 1	32
Gambar 4. 7. Testing pada gambar 2 skenario 1	32
Gambar 4. 8. Testing pada gambar 3 skenario 1	33
Gambar 4. 9 Testing pada gambar 1 skenario 2	34
Gambar 4. 10 Testing pada gambar 2 skenario 2	34
Gambar 4. 11. Testing pada gambar 3 skenario 2	35
Gambar 4. 12. Testing pada gambar 1 skenario 3	35
Gambar 4. 13. Testing pada gambar 2 skenario 3	36
Gambar 4. 14. Testing pada gambar 3 skenario 3	36
Gambar 4. 15. Testing pada gambar 1 skenario 4	37
Gambar 4. 16. Testing pada gambar 2 skenario 4	37
Gambar 4. 17. Testing pada gambar 3 skenario 4	37

BAB I

PENDAHULUAN

1.1 Latar Belakang

Keselamatan dalam berkendara adalah hal yang selalu diharapkan ketika seseorang sedang mengendarai ataupun menjadi penumpang moda transportasi. Namun, tak jarang banyak hal yang tidak diharapkan dapat terjadi tanpa pernah kita sangka dan kita duga terjadinya, seperti kecelakaan dalam berkendara. Menurut *World Health Organization* (WHO) pada tahun 2019 diperkirakan sekitar 1.35 juta orang di seluruh dunia meninggal karena kecelakaan lalu lintas dan sebanyak 20%-30% diakibatkan karena kelelahan (Sinha et al., 2021). Pada penelitian lain yang dilakukan oleh (Yang et al., 2021) disebutkan indikasi kelelahan pada pengemudi dapat ditandai dengan mata yang berkedip diikuti dengan anggukan kepala, mata yang sering menutup saat mengemudi tetapi disebutkan jika menguap adalah indikasi utama seorang pengemudi mulai mengalami kelelahan.

Distraksi saat mengemudi dapat juga menjadi salah satu pemicu kecelakaan (Yazdi & Soryani, 2019). Penggunaan *mobile phone* saat mengemudi merupakan salah satu distraksi karena menyebabkan pengemudi kehilangan fokus berkendara. Namun, disebutkan pada penelitian ini bahwa faktor utama kecelakaan berkendara tidak hanya distraksi melainkan kelelahan dan mengantuk yang dialami pengemudi. Mengantuk adalah keadaan dimana seseorang berada di fase terjaga dan tertidur (Chaabene et al., 2021). Pada keadaan mengantuk seseorang menjadi lambat dalam merespon sesuatu, refleks anggota badan tidak bekerja maksimal, dan kesulitan untuk menjaga posisi kepala untuk tetap tegak menjaga jarak pandang. Mengemudi dalam keadaan mengantuk merupakan salah satu tindakan berbahaya yang dapat menyebabkan ribuan kematian dan cedera di setiap tahunnya, hal ini disebabkan karena dalam keadaan mengantuk mengakibatkan penurunan keterampilan dalam mengemudi sehingga meningkatkan risiko kecelakaan (Higgins et al., 2017).

Beban kerja kognitif selama mengemudi juga dianggap sebagai salah satu penyebab kecelakaan yang dapat diidentifikasi dengan mendeteksi respon fisiologis pada pengemudi (Tjolleng et al., 2017). Mempertahankan tingkat beban kerja kognitif dalam kondisi aman menjadi hal yang krusial untuk memastikan bahwa pengemudi berada dalam kondisi yang optimal saat mengemudi. Ada banyak metode yang dilakukan untuk memantau beban kerja kognitif pada pengemudi diantaranya dengan melakukan pemantauan detak jantung dan identifikasi melalui ekspresi wajah. (Almogbel et al., 2018).

Pencegahan kecelakaan yang diakibatkan kantuk pada pengemudi dapat dilakukan dengan cara melakukan deteksi kantuk sejak dini dan secara akurat (Gwak et al., 2020). Deteksi kantuk merupakan salah satu topik penting dalam keselamatan berkendara yang bertujuan untuk mengurangi frekuensi dan tingkat keparahan kecelakaan lalu lintas (Ma et al., 2020). Langkah penting dalam mendeteksi kantuk adalah dengan melakukan deteksi wajah dan deteksi ekspresi. Penelitian terkait deteksi wajah untuk mendeteksi kantuk pada pengemudi pernah dilakukan oleh beberapa peneliti, seperti yang dilakukan oleh (Sinha et al., 2021) yang melakukan demonstrasi untuk menentukan apakah pengemudi mengantuk atau tidak dengan menganalisis daerah mata dan mulut dengan kamera. Pada penelitiannya diharapkan agar dapat menggunakan kamera dengan kualitas lebih baik untuk mengambil gambar untuk meningkatkan kinerja sistem dan juga dapat disertakan penambahan audio pada setiap *video frame*.

Seiring berkembangnya teknologi, dalam beberapa dekade terakhir kecerdasan buatan untuk melakukan deteksi kantuk pada pengemudi menjadi subjek banyak penelitian dan berbagai metode telah dikembangkan (Faraji et al., 2021). Deteksi kantuk merupakan salah satu implementasi dari deteksi objek (*object detection*) yang menggunakan teknik pembelajaran mendalam (*deep learning*). Tujuan dari deteksi objek adalah untuk menentukan apakah ada contoh objek dari kategori yang sudah ditentukan dalam gambar dengan menggunakan kotak pembatas (*bounding box*) untuk menentukan luas dari setiap objek yang terdeteksi (Liu et al., 2020).

Sebuah penelitian yang membandingkan kinerja algoritma SVM (*Support Vector Machine*), HMM (*Hidden Markov Model*) dan CNN (*Convolutional Neural Network*) menyebutkan bahwa CNN mengungguli kedua metode yang lain untuk melakukan deteksi kantuk pada pengemudi (Ngxande et al., 2017). CNN adalah salah satu metode *machine learning* dari pengembangan *multi layer perceptron* (MLP) yang di desain untuk

mengolah data dua dimensi. CNN termasuk dalam jenis *deep neural network* karena dalamnya tingkat jaringan dan banyak diimplementasikan dalam data citra (Mawan, 2020). Beberapa penelitian yang menggunakan arsitektur CNN untuk melakukan deteksi kantuk dilakukan oleh (Yazdi & Soryani, 2019) mendeteksi kantuk berdasarkan pengolahan citra apakah mata terbuka atau tertutup. Penelitian yang dilakukan (Faraji et al., 2021) melakukan ekstraksi fitur wajah menggunakan CNN, *you only look once* (YOLO), dan menggunakan *long short term memory* (LSTM) untuk menghitung periode waktu menguap dan berkedip. Penelitian yang dilakukan oleh (Sinha et al., 2021) membandingkan algoritma *viola jones*, DLib, dan YOLO untuk melakukan deteksi kantuk dan diperoleh kesimpulan bahwa YOLO dapat melakukan deteksi lebih akurat daripada *viola jones* dan DLib. Berdasarkan uraian permasalahan di atas, maka metode yang dapat digunakan untuk perancangan sistem deteksi kantuk adalah menggunakan YOLO.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah bagaimana mengimplementasikan algoritma *deep learning you only look once* untuk deteksi kantuk pada pengemudi.

1.3 Batasan Masalah

Batasan penelitian yang diberikan agar lebih terarah dan sesuai dengan yang dimaksudkan sebagai berikut :

1. Software yang digunakan untuk mendeteksi deteksi kantuk adalah *google colab*
2. Dataset yang digunakan untuk melatih sistem diambil dari kaggle dan github
3. Klasifikasi terbagi menjadi 2 kelas, yaitu menguap (*drowsy*) dan terjaga (*awake*)
4. Data latih dan data uji dalam format gambar.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah mengimplementasikan algoritma *deep learning you only look once* untuk deteksi kantuk pada pengemudi.

1.5 Manfaat Penelitian

Manfaat penelitian ini sebagai berikut :

1. Dapat dimanfaatkan dalam pengembangan teknologi sistem deteksi untuk keselamatan pengemudi sebagai peringatan pengemudi sedang dalam keadaan mengantuk atau tidak
2. Hasil penelitian dapat dikembangkan menjadi teknologi baru yang dapat diterapkan di Indonesia untuk mengurangi angka kecelakaan yang disebabkan kantuk saat mengemudi

1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir disusun sebagai berikut:

BAB I PENDAHULUAN

Memberikan penjelasan terkait latar belakang permasalahan, rumusan masalah, batasan permasalahan, tujuan penelitian, manfaat penelitian, dan sistematika penelitian.

BAB II KAJIAN LITERATUR

Menguraikan tentang konsep dan penjelasan metode yang digunakan. Selain itu terdapat hasil-hasil penelitian yang telah dilakukan sebelumnya oleh peneliti lain yang terkait dengan penelitian yang dilakukan.

BAB III METODE PENELITIAN

Menjelaskan terkait alur penelitian, teknik yang digunakan serta data yang akan dikaji.

BAB IV PENGUMPULAN DAN PENGOLAHAN DATA

Menguraikan tentang data yang diperoleh yang ditampilkan dalam bentuk tabel serta cara menganalisa data tersebut. Pengolahan data termasuk analisis yang dilakukan terhadap data yang diperoleh.

BAB V PEMBAHASAN

Memaparkan pembahasan berupa bagaimana algoritma deep learning yolo dapat mendeteksi kantuk pada pengemudi serta evaluasi terhadap sistem yang sudah dibuat.

BAB VI KESIMPULAN DAN REKOMENDASI

Menjelaskan tentang kesimpulan terhadap hasil penelitian yang telah dilakukan. Selain itu diberikan juga rekomendasi yang dapat digunakan oleh penelitian selanjutnya berdasarkan hasil yang sudah dicapai.



BAB II

KAJIAN LITERATUR

Kajian literatur terdiri dari kajian deduktif dan kajian induktif. Kajian deduktif berisi penjelasan landasan teori yang berkaitan dengan penelitian. Pada penelitian kali ini landasan teori mencakup *artificial intelligence*, *machine learning*, *deep learning*, *convolutional neural network*, *you only look once*, dan *object detection*. Selanjutnya pada kajian induktif berisi tentang beberapa penelitian sebelumnya yang mendukung penelitian ini.

2.1 Kajian Deduktif

2.1.1 *Artificial Intelligence*

Artificial Intelligence atau kecerdasan buatan didefinisikan sebagai kemampuan sistem untuk menginterpretasikan data eksternal dengan tepat untuk belajar dari data tersebut, dan menggunakan pembelajaran tersebut untuk mencapai tujuan dan tugas tertentu melalui adaptasi yang fleksibel (Haenlein & Kaplan, 2019).

2.1.2 *Machine Learning*

Machine learning adalah cabang dari *artificial intelligence*. Kecerdasan buatan memiliki pengertian yang sangat luas tapi secara umum dapat dipahami sebagai komputer dengan kecerdasan layaknya manusia. *Machine learning* atau pembelajaran mesin adalah studi ilmiah tentang algoritma dan model statistik yang digunakan sistem komputer untuk melakukan tugas tertentu tanpa diprogram secara eksplisit (Mahesh, 2018).

2.1.3 *Deep Learning*

Deep learning atau pembelajaran mendalam adalah cabang *machine learning* dengan algoritma jaringan syaraf tiruan yang dapat belajar dan beradaptasi terhadap sejumlah besar data. Pembelajaran mendalam memungkinkan model komputasi dari beberapa lapisan pemrosesan untuk mempelajari dan mewakili data dengan berbagai tingkat abstraksi yang meniru bagaimana otak memahami dan memahami informasi multimodal, sehingga secara implisit menangkap struktur rumit dari data skala besar (Voulodimos et al., 2018).

2.1.4 *Computer Vision*

Computer vision atau visi komputer dapat didefinisikan dengan pengertian pengolahan citra yang dikaitkan dengan akuisisi citra, pemrosesan, klasifikasi, penganan, dan pencakupan keseluruhan, pengambilan keputusan yang diikuti pengidentifikasian citra. Inti dari teknologi *computer vision* adalah untuk menduplikasi kemampuan penglihatan manusia ke dalam benda elektronik sehingga benda elektronik dapat memahami dan mengerti arti dari gambar yang dimasukkan (Prabowo & Abdullah, 2018).

2.1.5 *Object Detection*

Deteksi objek adalah proses mendeteksi objek semantik dari kelas tertentu (seperti manusia, pesawat terbang, atau burung) dalam gambar digital dan video. Pendekatan umum untuk kerangka kerja deteksi objek mencakup pemrosesan sekumpulan dataset yang diklasifikasikan menggunakan fitur CNN (Voulodimos et al., 2018).

Deteksi objek menjadi salah satu penelitian yang paling penting pada *computer vision*. Algoritma deteksi objek dibagi menjadi dua, yaitu metode tradisional (*traditional methods*) dan metode pembelajaran mendalam (*deep learning methods*). Pada *deep learning methods*, algoritma deteksi objek dikelompokkan menjadi dua, yaitu berdasarkan pada algoritma deteksi objek proposal wilayah (*region proposal object detection algorithms*), termasuk RCNN, SPP-net, Fast-RCNN dan Faster-RCNN. Kemudian yang kedua berdasarkan pada algoritma deteksi objek regresi (*regression object detection algorithm*), termasuk SSD dan YOLO (Liu et al., 2018).



Gambar 2. 1. Contoh deteksi pada objek

Sumber : (Voulodimos et al., 2018)

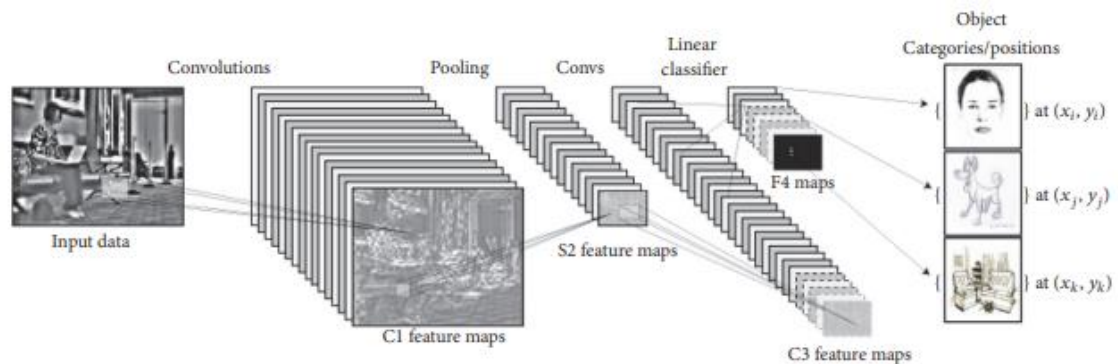
2.1.6 Convolutional Neural Network (CNN)

Dilihat dari arsitekturnya, *Convolutional Neural Networks* (CNN) termasuk ke dalam kelas *deep feed-forward artificial neural networks* yang banyak diaplikasikan pada analisis citra. Arsitektur CNN sangat sederhana yaitu terdiri atas satu lapis masukan (*input layer*), suatu lapis keluaran (*output layer*), dan sejumlah lapis tersembunyi (*hidden layers*) (Alom et al., 2018).

CNN terdiri dari tiga jenis utama lapisan saraf, yaitu *convolutional layers*, *pooling layers*, dan *fully connected layers*. Setiap jenis lapisan memainkan peran yang berbeda. Gambar 2.2 menunjukkan arsitektur CNN untuk deteksi objek pada *computer vision*. CNN telah sukses diaplikasikan pada *computer vision*, seperti pengenalan wajah (*face recognition*), deteksi objek (*object detection*), robotika, dan *self-driving cars* (Voulodimos et al., 2018). Secara umum, layer pada CNN dibedakan menjadi dua yaitu layer ekstraksi fitur gambar dan layer klasifikasi (Khairunnas et al., 2021).

1. Layer ekstraksi fitur gambar, letaknya berada pada awal arsitektur tersusun atas beberapa layer dan setiap layer tersusun atas neuron yang terkoneksi pada daerah lokal (*local region*) layer sebelumnya. Layer jenis pertama adalah layer konvolusi dan layer kedua adalah layer *pooling*. Setiap layer diberlakukan fungsi aktivasi. Posisinya berselang-seling antara jenis pertama dengan jenis kedua. Layer ini menerima input gambar secara langsung dan memprosesnya.

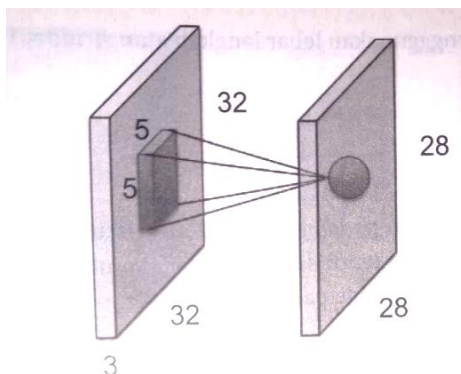
2. Layer klasifikasi, tersusun atas beberapa layer dan setiap layer tersusun atas neuron yang terkoneksi secara penuh (*fully connected*) dengan layer lainnya. Layer ini menerima input dari hasil keluaran layer ekstraksi fitur gambar berupa vektor kemudian ditransformasikan seperti *Multi Neural Networks* dengan tambahan beberapa *hidden layer*. Hasil keluaran berupa skoring kelas untuk klasifikasi.



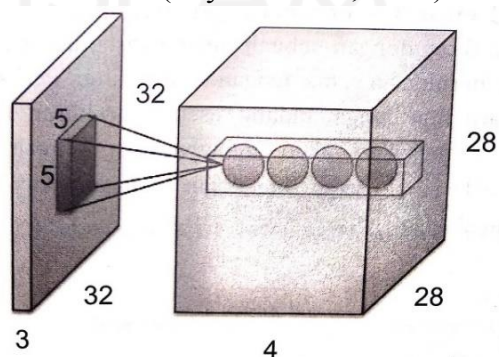
Gambar 2. 2. Arsitektur CNN untuk *computer vision*
 Sumber : (Voulodimos et al., 2018)

2.1.7 Convolutional Layers

Convolutional layer atau lapis konvolusional merupakan blok bangunan inti CNN, di mana sebagian besar komputasi dilakukan di lapis ini. Misalkan, kita membangun sebuah *convolutional layer* dengan satu lembaran neuron yang berisi 28×28 . Setiap lembar terhubung dengan suatu area kecil dalam (citra) masukan, misalnya 5×5 (piksel), yang merupakan bidang reseptif (*receptive field*) untuk setiap neuron dan menyatakan bahwa filter yang digunakan berukuran 5×5 , seperti diilustrasikan pada gambar 2.3 dan 2.4. Seluruh bidang reseptif akan ditelusuri secara tumpang tindih parsial, maka semua neuron tersebut pasti berbagi bobot koneksi atau *weight sharing* (Suyanto et al., 2019).



Gambar 2. 3. *Convolution layer* dengan satu buah filter berukuran 5 x 5
Sumber : (Suyanto et al., 2019)

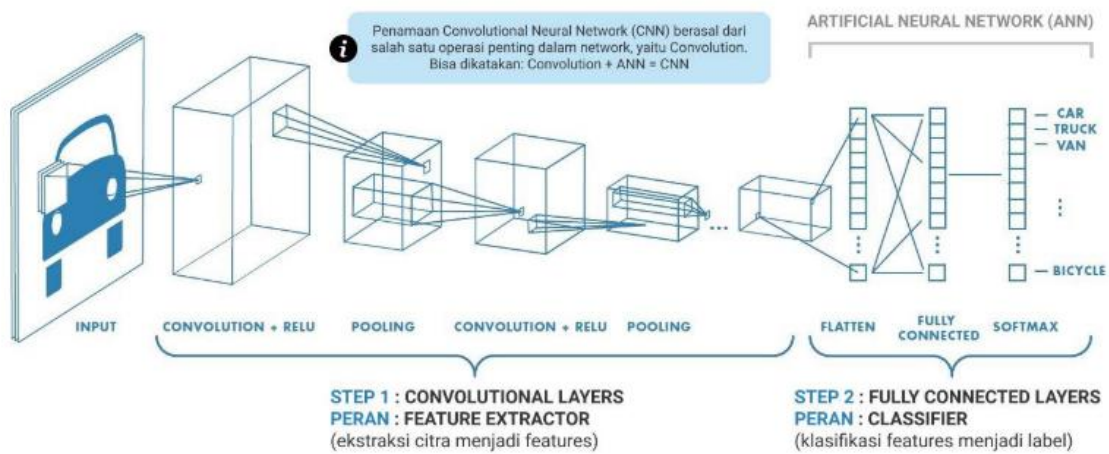


Gambar 2. 4. *Convolution layer* dengan empat buah filter berukuran 5 x 5
Sumber : (Suyanto et al., 2019)

2.1.8 *Fully Connected Layers*

Pada lapisan yang terhubung secara penuh atau *fully connected layer*, setiap neurons memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya. Hal ini sama persis dengan yang ada pada *Multi Layer Perceptron* (MLP). Model aktivasinya pun sama persis dengan MLP, yaitu komputasi menggunakan suatu perkalian matriks yang diikuti dengan bias offset (Suyanto et al., 2019).

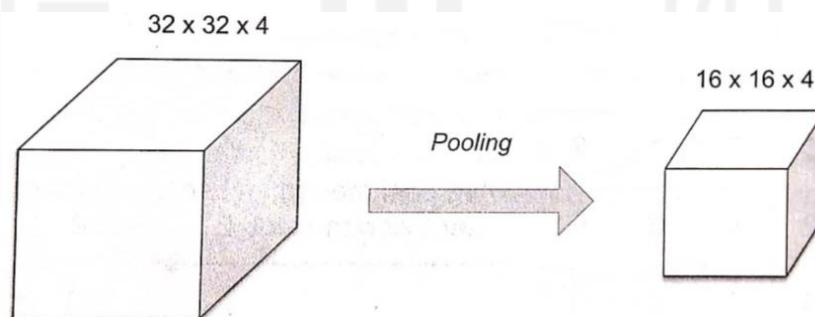
Pada lapisan ini terdapat hidden layer yang berfungsi sebagai sebagai fungsi aktivasi, pada umumnya digunakan *Rectified Linear Unit* (ReLU) dan pada output layer terdapat juga fungsi aktivasi, pada kasus klasifikasi 2 kelas pada umumnya digunakan fungsi aktivasi softmax. Tujuan utama dari fully connected layer adalah mengolah data sehingga dapat dilakukan klasifikasi. Output dari fully connected layer berupa nilai probabilitas terhadap kelas (Jessar et al., 2021).



Gambar 2. 5. Lapisan *fully connected layer*
Sumber : (Jessar et al., 2021)

2.1.9 Pooling Layers

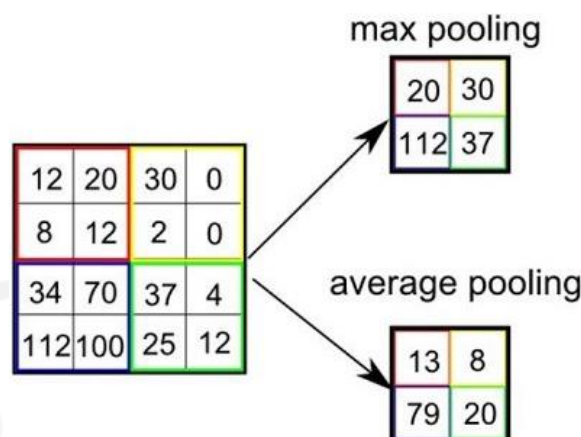
Pooling layer berfungsi menjaga ukuran data ketika *convolution* dilakukan, yaitu dengan melakukan reduksi sampel (*downsampling*), seperti diilustrasikan pada gambar 2.5. Dengan *pooling*, kita dapat merepresentasikan data menjadi lebih kecil, mudah dikelola, dan mudah mengontrol *overfitting*.



Gambar 2. 6. Proses *pooling* untuk mereduksi dimensi data
Sumber : (Suyanto et al., 2019)

Pada umumnya, proses pooling dilakukan menggunakan *max pooling* atau *average pooling*. Caranya adalah dengan memilih nilai maksimum atau nilai rata-rata dalam suatu area tertentu, seperti diilustrasikan pada gambar 2.6. Sementara itu, pada gambar *average pooling*, dari empat nilai di bagian kiri atas akan dihasilkan satu nilai rata-rata. Pada banyak permasalahan praktis, teknik *max pooling* umumnya memberikan

performansi yang lebih baik dibanding *average pooling*, *L2-norm pooling*, atau teknik yang lain (Suyanto et al., 2019).



Gambar 2. 7. Teknik *max pooling* dan *average pooling*
Sumber : (Saha, 2018)

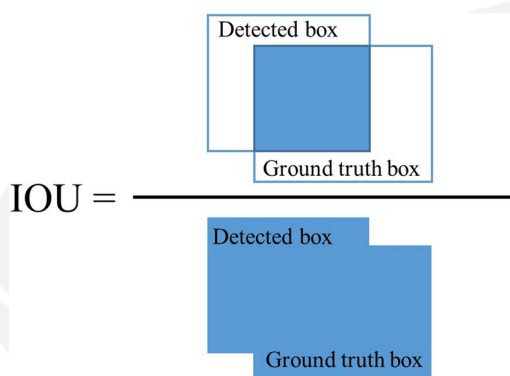
2.1.10 *You Only Look Once (YOLO)*

Algoritma YOLO adalah salah satu algoritma yang digunakan untuk melakukan deteksi pada objek yang pertama kali dikenalkan oleh Joseph Redmon dengan menggunakan *custom framework* bernama darknet. Algoritma YOLO menggunakan jaringan saraf konvolusi tunggal (*single convolution neural network*) yang memiliki kelebihan kecepatan deteksi yang tinggi. YOLO menggunakan jaringan saraf konvolusi khusus untuk melakukan klasifikasi dan lokasi beberapa objek dalam satu gambar sekaligus. Hal ini tentu saja mengorbankan beberapa akurasi, tetapi masih lebih akurat jika dibandingkan dengan algoritma pendeteksian objek lain seperti R-CNN (Tao et al., 2018).

YOLO mengintegrasikan kotak kandidat ekstraksi (*candidate boxes extraction*), ekstraksi fitur (*feature extraction*), dan klasifikasi objek ke dalam jaringan syaraf. YOLO secara langsung mengekstrak *candidate boxes* dari gambar dan objek yang terdeteksi melalui seluruh fitur gambar. Di jaringan YOLO, gambar dibagi menjadi berukuran $s \times s$ dengan *bounding box* terdistribusi secara merata pada sumbu X dan sumbu Y. Jaringan ini membagi gambar ke dalam beberapa *region* dan melakukan prediksi berdasarkan lokasi pada setiap *bounding boxes*, *confidence value*, dan probabilitas kelas untuk setiap *region* (Liu et al., 2018).

Gambar yang sudah menjadi terbagi menjadi berukuran $s \times s$, pada setiap sel kisi akan memprediksi *bounding box* dan *confidence score* (Redmon et al., 2016). *Confidence*

score menunjukkan tingkat keyakinan model dalam melakukan deteksi dan mengukur seberapa besar akurasi sebuah objek yang terdeteksi pada *bounding box*. Setiap *bounding box* terdiri atas 5 prediksi nilai yaitu x , y , w , h , dan *confidence*. Koordinat (x,y) merupakan pusat dari kotak ke batas sel grid. Koordinat (w,h) merupakan lebar dan tinggi dari keseluruhan gambar. Sedangkan *confidence score* mewakili *intersection over union* (IoU) antara kotak prediksi (*predicted box*) dan luas objek yang sebenarnya pada gambar (*ground truth box*). IoU adalah metrik evaluasi yang digunakan untuk mengukur keakuratan detektor objek berdasarkan dataset yang sudah di latih sebelumnya dan diilustrasikan seperti gambar 2.8.



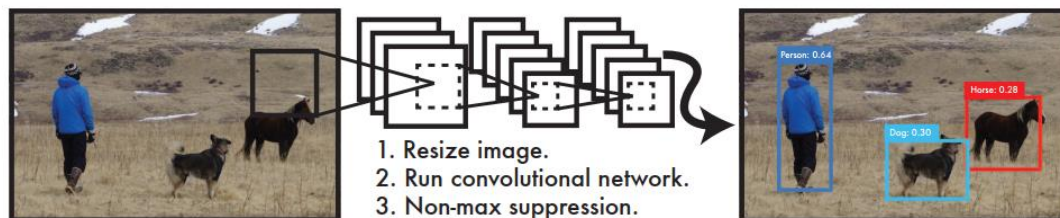
Gambar 2. 8. *Intersection over union*
Sumber : (Mahdi et al., 2020)

Jika tidak ada objek yang terdeteksi pada *bounding box* maka nilai *confidence score* adalah 0. Class confidence score mengukur nilai kepercayaan pada setiap bounding box yang memberikan kode kemungkinan kelas yang muncul dalam kotak dan berapa nilai prediksinya. Untuk menghitung *class confidence score* dapat dirumuskan sebagai berikut (Redmon et al., 2016) :

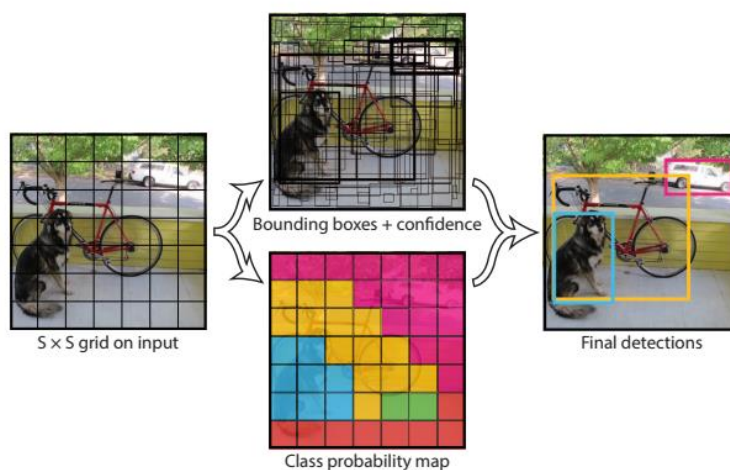
$$\text{Confidence score} = \text{Pr}(\text{Class}) * \text{IoU}_{pred}^{\text{truth}} \quad (2.1)$$

YOLOv4 merupakan versi terbaru yang dikembangkan oleh (Bochkovskiy et al., 2020) yang menambahkan sejumlah implementasi tambahan yang meningkatkan akurasi dan efisiensi pendeteksian. Arsitektur ini menggunakan arsitektur backbone CNN

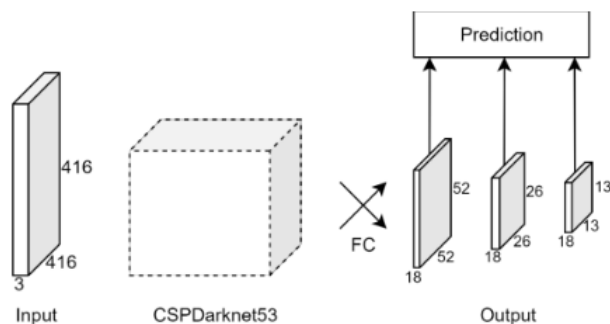
CSPDarknet53. Arsitektur ini berisi 162 lapisan dengan input gambar yang sudah dilakukan konfigurasi sesuai penelitian.



Gambar 2. 9. Deteksi objek pada sistem menggunakan YOLO
(Sumber : Redmon et al., 2016)



Gambar 2. 10. Proses deteksi objek menggunakan YOLO
(Sumber : Redmon et al., 2016)



Gambar 2. 11. Arsitektur jaringan pada YOLOv4
(Sumber : Dhiaegana, 2020)

2.1.11 Google Colaboratory

Google Colaboratory atau yang biasa disingkat "Colab" adalah *coding environment* bahasa pemrograman python dengan format "notebook" (mirip dengan *Jupyter notebook*)

yang dapat di akses menggunakan browser untuk membuat program atau melakukan pengolahan dengan beberapa keuntungan yaitu tidak memerlukan konfigurasi, akses gratis ke GPU, dan berbasis cloud. Pada *google colab* digunakan GPU Nvidia Tesla T4.

2.2 Kajian Induktif

Kajian induktif merupakan hasil penelitian-penelitian yang telah dilakukan oleh peneliti sebelumnya. Pada penelitian ini diambil penelitian dengan rentang tahun 2020-2021. Berikut merupakan rangkuman dari beberapa penelitian yang sudah dilakukan dan pada tabel 2.1 berisikan tabel perbandingan metode dan objek penelitian tiap penelitian.

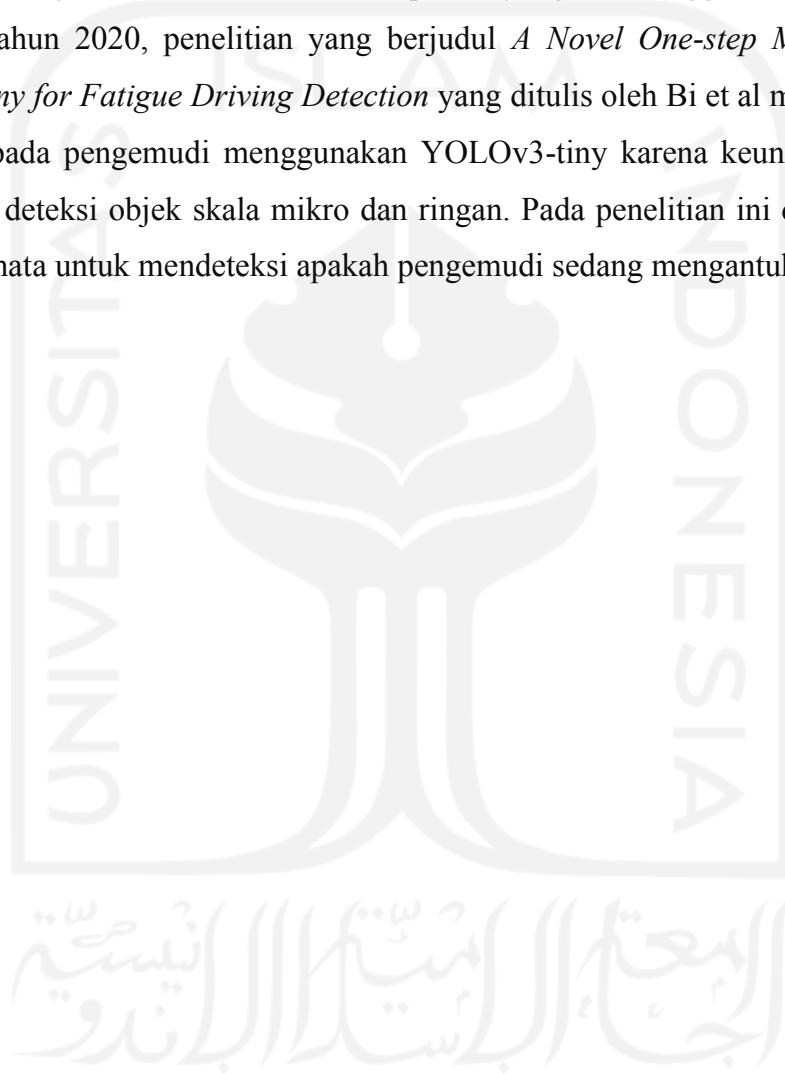
Pada penelitian yang berjudul *Drowsiness Detection System Using Deep Learning* yang dilakukan oleh Sinha et al (2021) melakukan deteksi kantuk pada pengemudi dengan menggunakan area deteksi pada wajah yaitu mata dan mulut. Tujuan dari penelitian ini untuk membandingkan algoritma deteksi objek mana yang lebih baik saat melakukan deteksi. Hasil penelitian yang diperoleh pada penelitian ini menunjukkan jika YOLOv3 memiliki arsitektur deteksi objek lebih baik dibandingkan dengan Viola Jones dan DLib. Sistem dapat mendeteksi dengan akurasi 97% untuk kecepatan 20fps.

Penelitian lain tentang deteksi kantuk pada pengemudi juga dilakukan oleh Biju & Edison (2020) dengan judul *Drowsy Driver Detection Using Two Stage Convolutional Neural Networks*. Tujuan dari penelitian ini yaitu dapat memproses input wajah secara real time secara efisien dan hemat biaya dengan menggunakan YOLOv3. Berdasarkan penelitian yang sudah dilakukan metodologi yang diusulkan memberikan akurasi masing masing 80.32%, 79.34% dan 89.90% pada tiga jenis input gambar yang berbeda.

Penggunaan YOLOv3 untuk melakukan deteksi kantuk juga dilakukan oleh Faraji et al (2021) yang berjudul *Drowsiness Detection Based on Driver Temporal Behavior Using a New Developed Dataset* yang melakukan deteksi dengan menggabungkan LSTM dan YOLOv3. Tujuan dari penelitian ini yaitu untuk menunjukkan efektivitas apakah penggabungan CNN dengan LSTM dapat melakuakn deteksi lebih baik. Hasil yang diperoleh pada penelitian ini yakni CNN YOLOv3 digunakan sebagai untuk mendeteksi wajah dan LSTM diterapkan untuk klasifikasi dari objek yang terdeteksi. Sistem dapat melakukan deteksi dengan akurasi sebesar 91.7%.

Penelitian tentang deteksi kantuk dengan mengukur frekuensi pengemudi saat memejamkan mata, frekuensi menguap, frekuensi mata menutup dalam mengidentifikasi kantuk pada pengemudi juga dilakukan oleh Pan (2020) pada penelitiannya yang berjudul *The Drowsy Testing System According to Deep-Learning*. Menggunakan backbone algoritma YOLOv3 pada proses deteksinya, penelitian ini mengembangkan algoritma yang diberi nama YOLO FD (Fatigue Detection) yang menunjukkan bahwa algoritma yang dikembangkan memiliki akurasi dan presisi yang lebih tinggi.

Pada tahun 2020, penelitian yang berjudul *A Novel One-step Method Based on YOLOv3-tiny for Fatigue Driving Detection* yang ditulis oleh Bi et al melakukan deteksi kelelahan pada pengemudi menggunakan YOLOv3-tiny karena keunggulannya dalam melakukan deteksi objek skala mikro dan ringan. Pada penelitian ini deteksi dilakukan pada area mata untuk mendeteksi apakah pengemudi sedang mengantuk.



Tabel 2. 1. Tabel kajian induktif

No	Penulis	Metode				Objek Penelitian	Area Deteksi
		Viola Jones	DLib	LSTM	YOLOv3		
1	Avigyan Sinha, R. P. Aneesh, Sarada K. Gopal (2021)	✓	✓		✓	Dataset pengemudi yang dikumpulkan dari kamera Near-Infrared (NIR)	Wajah
2	Aishwarya Biju, Anitha Edison (2020)				✓	Dataset Closed Eyes in the Wild (CEW) dan dataset National Tsuing Hua University (NTHU) Driver Drowsiness Detection	Wajah
3	F. Faraji, F. Lotfi, J. Khorramdel, A. Najafi, A. Ghaffari (2021)			✓	✓	Dataset pengemudi yang diperoleh menggunakan kamera PS3	Wajah
4	Pan Jiasheng (2020)				✓	Dataset operator yang dikumpulkan dari kamera	Mata dan mulut
5	Xiaojun Bi, Zheng Chen, Jianyu Yue (2020)				✓	Dataset Closed Eyes in the Wild (CEW) dan <i>real driving dataset</i>	Mata

BAB III

METODE PENELITIAN

3.1 Objek Penelitian

Objek penelitian pada penelitian ini adalah gambar pengemudi yang mengantuk (*drowsy*) dan terjaga (*awake*) untuk dilatih pada sistem agar dapat digunakan untuk mendeteksi kantuk pada pengemudi.

3.2 Metode Pengumpulan Data

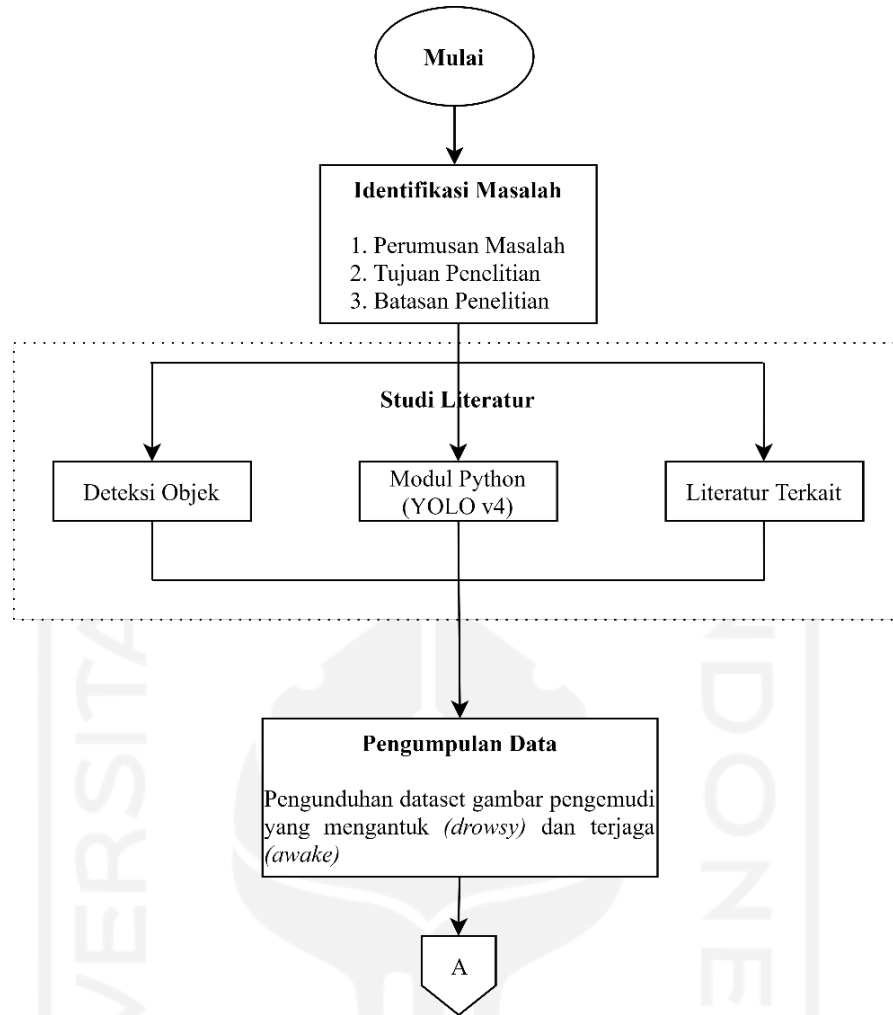
Metode pengumpulan data yang digunakan adalah dengan mengunduh data *open source* pada situs *kaggle* dan *github*.

3.3 Sumber Data

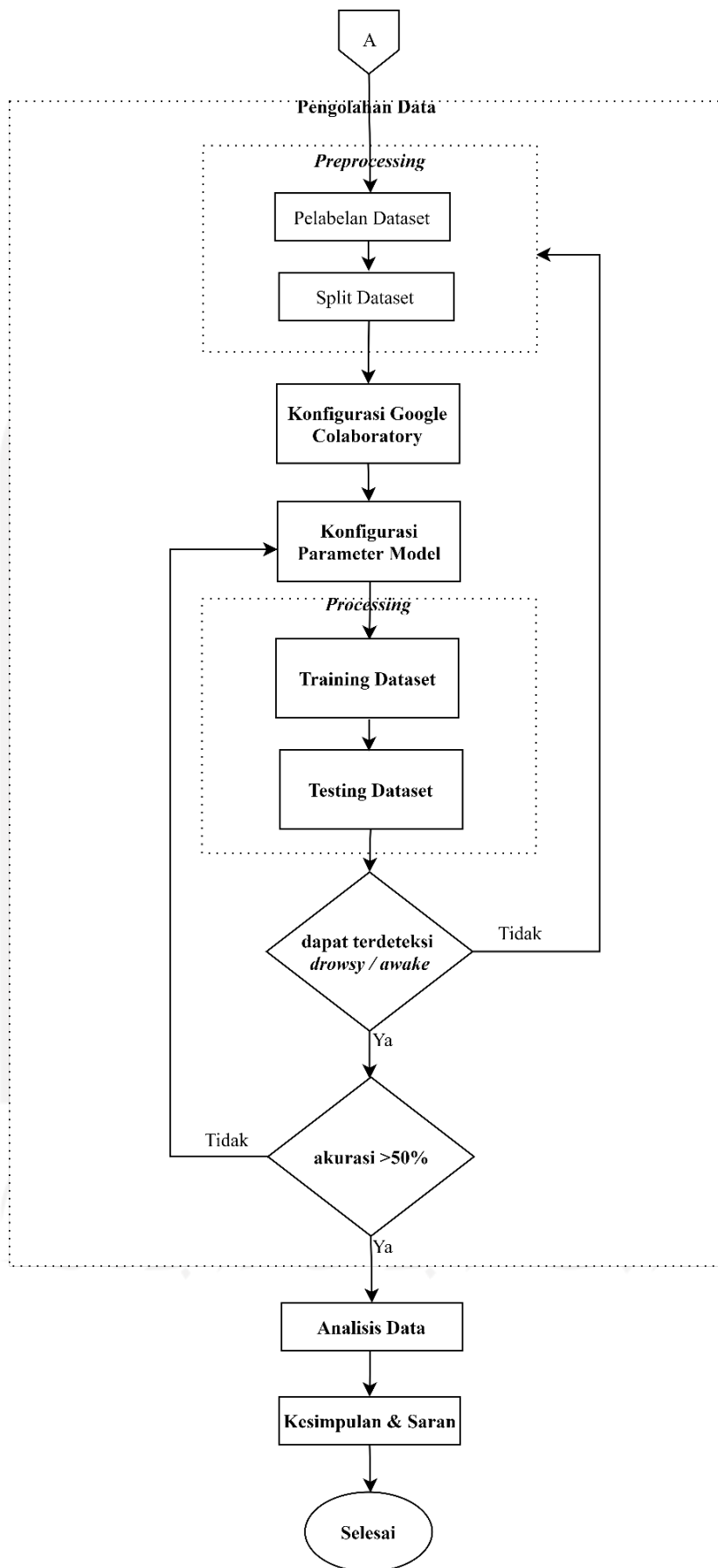
Berdasarkan sumbernya, data terbagi menjadi data primer dan data sekunder. Sumber data pada penelitian ini menggunakan data sekunder. Pada penelitian ini data sekunder yang digunakan yaitu dataset gambar pengemudi yang mengantuk dan terjaga yang diperoleh dari *website* kumpulan dataset yaitu *kaggle* dan *github*.

3.4 Alur Penelitian

Alur penelitian yang dilakukan pada penelitian ini dijelaskan pada gambar berikut. Gambar 3.1 dan 3.2 menunjukkan alur penelitian yang dilakukan pada penelitian ini. Pada gambar 3.1 menunjukkan proses identifikasi masalah hingga proses pengumpulan data dan selanjutnya pada gambar 3.2 menunjukkan proses lanjutan dari diagram sebelumnya yang menunjukkan proses pengolahan data hingga kesimpulan.



Gambar 3. 1. Diagram Alir Penelitian



Gambar 3. 2 Diagram Alir Penelitian Lanjutan

3.4.1 Identifikasi Masalah

Identifikasi masalah pada penelitian menjadi tahap awal untuk mengenalkan permasalahan yang akan diangkat dan ditentukan apa tujuan dari penelitian yang dilakukan berdasarkan permasalahan yang ada.

3.4.2 Studi Literatur

Pada tahap ini studi literatur digunakan sebagai referensi penelitian baik menggunakan jurnal yang melakukan penelitian sebelumnya ataupun menggunakan buku yang sesuai dengan penelitian ini yaitu tentang deteksi objek.

3.4.3 Pengumpulan Data

Proses pengumpulan data pada penelitian ini dilakukan dengan mengunduh dataset menggunakan data *open source* pada situs *kaggle* dan *github*.

3.4.4 Pengolahan Data

Setelah melakukan tahap pengumpulan data, kemudian data tersebut dilakukan olah data yang diawali dengan tahap *pre-processing* data dimana dataset diberi label dan dilakukan split dataset. Setelah tahap *pre-processing* dilakukan konfigurasi google colaboratory dan konfigurasi parameter model untuk melakukan training data dan testing data.

3.4.5 Analisis Data

Proses analisis data dapat dilakukan apabila model yang sudah dibuat dapat mendeteksi apakah pengemudi mengantuk (*drowsy*) atau terjaga (*awake*) saat dilakukan testing data dengan melakukan input sebuah gambar ke dalam model. Apabila model belum dapat melakukan deteksi maka dilakukan ulang tahap *pre-processing* data. Apabila model dapat melakukan deteksi dan akurasi yang diperoleh kurang dari 50% maka dilakukan

konfigurasi parameter model dan apabila akurasi yang di dapat sudah lebih dari 50% maka akan dilakukan analisis data sesuai pengolahan data yang sudah dilakukan.

3.4.6 Kesimpulan dan Saran

Tahap terakhir pada penelitian ini ditutup dengan menuliskan kesimpulan dari proses yang dilakukan dengan menjawab rumusan masalah yang sudah dituliskan. Selain diberikan juga saran yang dapat menjadi referensi untuk penelitian selanjutnya.



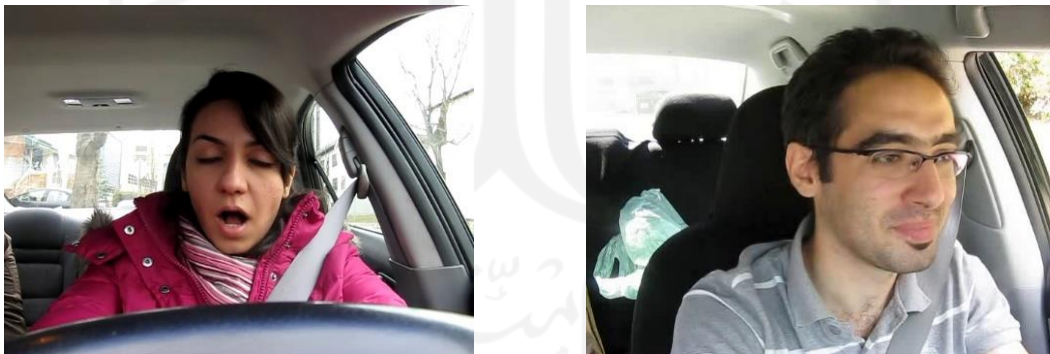
BAB IV

PENGUMPULAN DAN PENGOLAHAN DATA

4.1 Pengumpulan Dataset

Pada penelitian ini data yang diperoleh dari dataset yang tersedia pada data *open source* yaitu [kaggle](#). Data pada penelitian kali ini menggunakan 2 jenis gambar yaitu gambar pengemudi yang mengantuk (*drowsy*) dan terjaga (*awake*). Format gambar yang digunakan adalah .jpg dengan resolusi 640 x 480 p.

Gambar yang diambil ini akan digunakan untuk melatih model agar mengenali bagaimana pengemudi yang mengantuk dan pengemudi yang terjaga. Gambar yang nanti akan dilabeli akan diolah pada proses training dan testing yang akan menghasilkan model yang dapat mendeteksi wajah pengemudi yang mengantuk. Terdapat 200 gambar yang dikumpulkan dan dibagi menjadi 2 kelompok yaitu data training dan data testing.



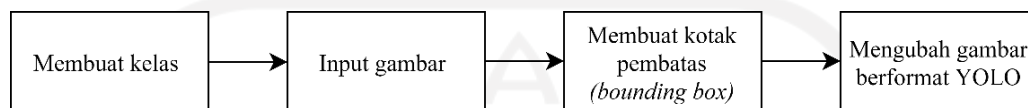
Gambar 4. 1. Gambar wajah mengantuk (*drowsy*) dan terjaga (*awake*)

4.2 *Pre-processing* Data

Pada tahap *pre-processing* data dilakukan proses pelabelan dataset dan *split* dataset. Pelabelan dataset digunakan untuk memberi label pada dataset yang sudah dikumpulkan, sedangkan *split* dataset digunakan untuk melakukan pembagian persentase *data training* dan *data testing*.

4.2.1 Pelabelan Dataset

Pada tahap ini dataset diberi label mengantuk (*drowsy*) atau terjaga (*awake*) menggunakan aplikasi LabelImg. Output dari pelabelan ini adalah gambar yang terlabeli dengan format YOLO dan tersimpan dalam tipe file .txt. Berikut merupakan diagram alir proses pelabelan gambar.



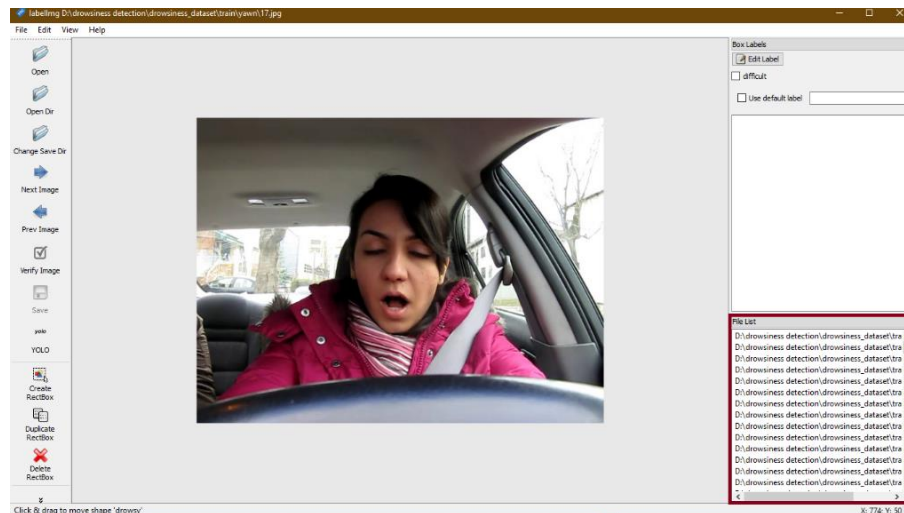
Gambar 4. 2. Diagram proses pelabelan dataset

4.2.1.1 Membuat Kelas

Seperti disebutkan pada poin 4.1 sebelumnya, pada perancangan deteksi kantuk ini menggunakan 2 kelas yaitu mengantuk (*drowsy*) dan terjaga (*awake*). File berformat names file ini nantinya akan digunakan untuk melakukan training data pada *google colaboratory*.

4.2.1.2 Input Gambar

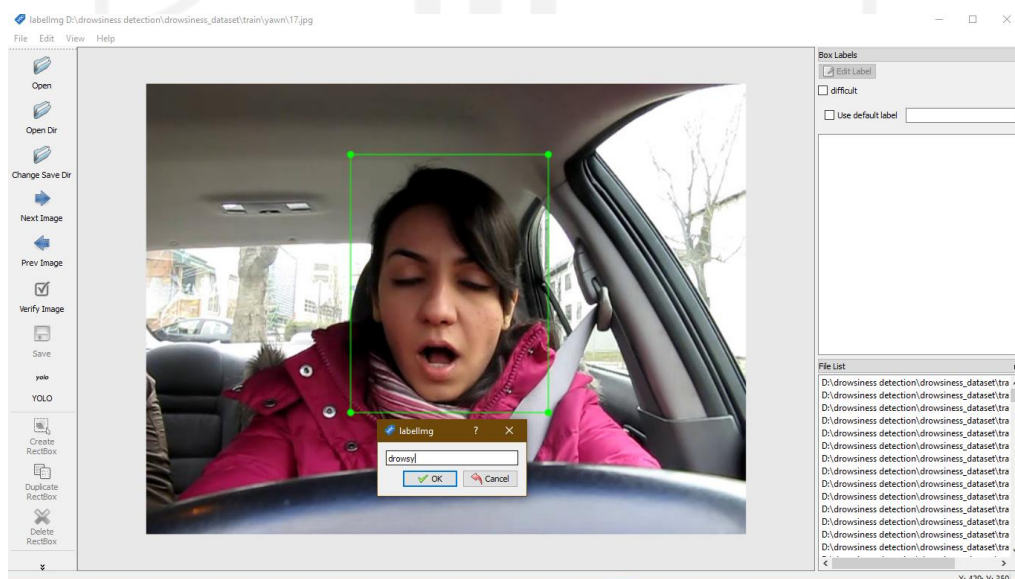
Langkah selanjutnya yaitu menggunakan aplikasi LabelImng untuk membuat kotak pembatas (*bounding box*). Namun sebelum membuat *bounding box*, terlebih dahulu membuka folder dataset yang sudah dikumpulkan.

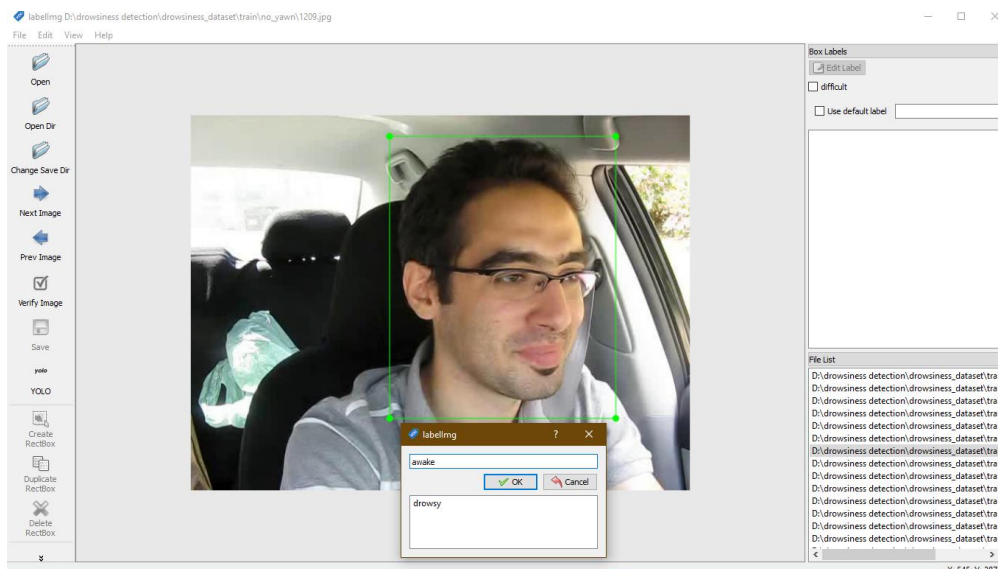


Gambar 4. 3. Proses input gambar

4.2.1.3 Membuat *Bounding Box*

Pembuatan *bounding box* dilakukan dengan menggunakan menu “*Create RectBox*” dan membuat kotak yang diarahkan pada gambar dan memberi label “*drowsy*” pada gambar pengemudi yang mengantuk dan label “*awake*” pada gambar pengemudi yang terjaga, seperti pada gambar 4.4 dan 4.5 di bawah.

Gambar 4. 4. Pembuatan *bounding box* untuk kelas mengantuk (*drowsy*)



Gambar 4. 5. Pembuatan *bounding box* untuk kelas terjaga (*awake*)

4.2.1.4 Mengubah Format Gambar

Setelah proses pembuatan *bounding box* tiap gambar, saat proses penyimpanan terdapat pilihan untuk menyimpan gambar dalam format Pascal/VOC atau YOLO. Pada penelitian ini digunakan format YOLO karena saat melakukan data training dan deteksi objek digunakan algoritma YOLO. Setelah gambar tersimpan dalam format YOLO akan muncul file .txt yang berisikan id indeks kelas, koordinat objek, tinggi dan lebar *bounding box*.

4.2.2 Split Dataset

Sebelum melakukan tahap training data, *split dataset* digunakan untuk melakukan pembagian persentase data *training* dan data *testing*. Pada penelitian kali ini akan dilakukan uji coba menggunakan 90:10 dimana 90% untuk data *training* dan 10% digunakan untuk data *testing* kemudian 80:20 dimana 80% untuk data training dan 20% untuk data testing sesuai prinsip pareto (Hemdan, n.d.). Pada penelitian ini total gambar yang digunakan sejumlah 200 gambar.

4.3 Konfigurasi *Google Colaboratory*

Pada tahap ini dilakukan *processing* data yang meliputi *training* dataset dan juga *testing* dataset. Namun, sebelum melakukan proses tersebut terlebih dahulu ada beberapa pengaturan yang harus dilakukan yaitu mengaktifkan runtime GPU untuk mempercepat proses *training* dataset yang memakan waktu cukup lama. Lalu yang kedua adalah melakukan aktivasi kerangka kerja (*framework*) darknet yang digunakan untuk melakukan konfigurasi parameter model.

4.4 Konfigurasi Parameter Model

Konfigurasi parameter model dilakukan untuk melakukan penyesuaian dengan pelatihan model yang akan dilakukan. Pada tabel 4.1 disajikan perubahan yang dilakukan pada parameter asli *yolov4-tiny* dan parameter *yolov4-tiny* yang disesuaikan dengan penelitian. Konfigurasi parameter model yang dilakukan merujuk pada pengembang yolov4 (Bochkovskiy et al., 2020) yang berjudul “*YOLO V4 : Optimal Speed and Accuracy of Object Detection*”. Konfigurasi yang dilakukan akan digunakan untuk melakukan 4 kali training data dengan rincian seperti di bawah ini.

Tabel 4. 1. Konfigurasi parameter model

No	Parameter	Training 1	Training 2	Training 3	Training 4
1	<i>Batch size</i>	64	64	64	64
2	<i>Network size</i>	416	416	416	416
3	<i>Subdivisions</i>	16	16	16	16
4	<i>Max batch</i>	4000	4000	4000	4000
5	<i>Filters</i>	21	21	21	21
6	<i>Class</i>	2	2	2	2
7	<i>Learning Rate</i>	0.00261	0.001	0.00261	0.001
8	<i>Split dataset</i>	90 : 10	90 : 10	80 : 20	80 : 20

4.4.1 *Batch size*

Dalam proses training, tidak dapat memasukan semua dataset sekaligus ke dalam *neural network*, oleh karena itu kita perlu membagi dataset ke dalam batch. Sama halnya dengan training dataset yang terdiri dari banyak gambar sehingga perlu dilakukan secara bertahap, sehingga dapat didefinisikan *batch size* adalah jumlah gambar per iterasi yang dilakukan saat melakukan training data.

Semakin besar *batch size* yang digunakan, maka kebutuhan komputasi pada GPU juga akan semakin besar. Terdapat beberapa nilai yang umum digunakan sebagai *batch size*, yaitu 16, 32, 64, 128, 256. Pada penelitian ini digunakan batch size 64 yang berarti akan memuat 64 gambar dalam satu kali iterasi, pemilihan batch ini disesuaikan dengan spesifikasi GPU digunakan yang tidak terlalu tinggi. Semakin tinggi nilai *batch* yang digunakan akan mempercepat proses training data namun beban kerja GPU juga semakin besar.

4.4.2 *Network size*

Network size adalah ukuran gambar yang akan dilatih pada model. Pada tahap pengumpulan gambar poin 4.1 di atas, gambar yang dikumpulkan memiliki ukuran 640x480 pixel akan di resize menjadi berukuran 416x416 pixel. Ukuran jaringan default dalam template konfigurasi pada umumnya menggunakan ukuran 416x416 pixel atau 608x608 pixel.

Semakin besar ukuran gambar yang digunakan maka proses training akan melambat dan lebih banyak memori GPU yang diperlukan, sedangkan semakin kecil ukuran gambar yang digunakan proses training menjadi lebih cepat namun sistem juga menjadi lebih sulit untuk mengenali objek karena ukuran yang terlalu kecil.

4.4.3 *Subdivisions*

Subdivisions digunakan untuk memecah *batch* menjadi *mini-batch* yang bertujuan untuk melakukan penyesuaian dengan memori GPU agar tidak terjadi *crash* pada runtime saat proses *training* data berlangsung. Jika menggunakan GPU dengan memori rendah, umumnya digunakan subdivisi 32 atau 64. Jika memiliki GPU dengan memori tinggi,

dapat digunakan subdivisi 16 atau 8, hal ini akan mempercepat proses pelatihan karena memuat lebih banyak gambar per iterasi.

Pada penelitian ini digunakan subdivisi 16 yang berarti 64 (*batch size*) dibagi dengan 16 (*subdivision*) yang berarti 4 gambar per mini-batch ini dikirim ke dalam *neural network* untuk diproses. Proses ini akan dilakukan 16 kali hingga batch selesai dan iterasi baru akan dimulai dengan 64 gambar baru.

4.4.4 *Max batch*

Max batch adalah jumlah maksimum iterasi yang dilatih pada jaringan. Persamaan yang digunakan untuk menghitung berapa *max batch* yang diperlukan adalah sebagai berikut (Bochkovskiy et al., 2020) :

$$\max_{batches} = \text{jumlah kelas} * 2000 \quad (4.1)$$

sehingga pada penelitian ini digunakan 4000 *max_batches* karena jumlah kelas yang digunakan sebanyak dua kelas. Semakin banyak nilai iterasi yang dilakukan maka proses training data akan semakin optimal namun waktu yang diperlukan semakin lama hal ini juga berpengaruh penggunaan memori GPU yang semakin banyak. Iterasi yang dilakukan tidak boleh memiliki nilai yang lebih rendah dari jumlah total gambar untuk training data.

4.4.5 *Filters*

Jumlah filter pada setiap lapisan sebelum konvolusi disesuaikan dengan jumlah kelas yang dilatih. Jumlah filter pada setiap lapisan konvolusi sebelum lapisan yolo diubah dengan menggunakan persamaan :

$$F = (C + Co + P) * M \quad (4.2)$$

dimana F sebagai jumlah filter, C sebagai jumlah kelas, Co sebagai jumlah koordinat yang digunakan (x, y, w, h), P sebagai nilai ke-objek-an suatu daerah yang diajukan, serta M sebagai jumlah anchor box yang digunakan (Dhiaegana, 2020). Sehingga pada penelitian ini jumlah filter diubah menjadi $(2+4+1)*3 = 21$.

4.4.6 *Class*

Kelas (*class*) atau label kelas yaitu variabel dependen dari model yang berupa kategori dan menjelaskan sebuah 'label' pada klasifikasi. Pada penelitian ini digunakan dua kelas yaitu mengantuk (*drowsy*) yang didefinisikan dengan *class_id* = 0 dan terjaga (*awake*) yang didefinisikan dengan *class_id* = 1

4.4.7 *Learning Rate*

Learning rate merupakan parameter yang digunakan untuk mengontrol seberapa cepat atau lambat model mempelajari masalah pada saat training. Jika *learning rate* terlalu kecil, diperlukan waktu yang lama untuk mencapai bobot yang optimal. Jika terlalu besar, bobot dengan nilai optimal tidak bisa diperoleh (Jessar et al., 2021). Pada penelitian kali ini nilai digunakan nilai learning rate 0.001 dan 0.00261 berdasarkan skenario training dataset yang sudah dilakukan pada poin 4.4 di atas.

4.5 *Processing Data*

Setelah melakukan semua konfigurasi parameter model dan *google colab* langkah selanjutnya adalah melakukan *training dataset* dan juga *testing dataset*. Output dari *training dataset* nantinya akan menunjukkan tingkat performansi model yang sudah dilakukan dengan melatih dataset yang sudah dilakukan pada tahap *pre-processing*. Kemudian pada *testing dataset* dilakukan dengan input berupa gambar untuk dilakukan deteksi pada model apakah dapat mengenali pengemudi yang mengantuk (*drowsy*) atau terjaga (*awake*).

4.5.1 *Training Dataset*

Pelatihan dataset digunakan agar model dapat digunakan untuk melakukan deteksi sesuai dengan konfigurasi yang sudah dilakukan pada poin 4.4. Pada penelitian ini, dilakukan skenario training dataset dengan melakukan perbandingan split dataset dan learning rate yang menghasilkan output nilai rata-rata IoU dan nilai loss dengan rincian pada tabel 4.2 di bawah.

Tabel 4. 2. Tabel skenario training dataset

No	Learning rate	Split Dataset	
		90% : 10%	80% : 20%
1	0.00261	Avg IoU : 78.15%	Avg IoU : 84.62%
		Avg Loss : 0.0286	Avg Loss : 0.0154
2	0.001	Avg IoU : 74.40%	Avg IoU : 79.14%
		Avg Loss : 0.0594%	Avg Loss : 0.0553

IoU merupakan metrik yang mengevaluasi keakuratan sistem dalam mendeteksi objek pada dataset yang telah dilatih. IoU membandingkan *ground-truth* atau objek pada citra dengan *predicted bounding box* dari model. Semakin tinggi atau besar nilai IoU yang diperoleh menunjukkan semakin dekat bounding box dari ground truth yang berarti nilai deteksi yang diperoleh juga semakin tinggi (Al et al., 2021). Sedangkan nilai loss merupakan suatu ukuran dari sebuah error yang dibuat oleh jaringan yang bertujuan untuk meminimalisir error tersebut. Semakin rendah atau kecil nilai loss yang diperoleh maka semakin sedikit kesalahan yang dilakukan model.

Berdasarkan training data yang sudah dilakukan dengan perbandingan seperti pada tabel 4.2 di atas, split data 80% : 20% dengan nilai learning rate 0.00261 memiliki nilai IoU paling tinggi dan nilai loss paling rendah daripada training dataset yang lain. Hal ini berarti pada penelitian kali ini melakukan deteksi dengan melakukan konfigurasi parameter menggunakan split dataset dengan nilai learning tersebut dapat menghasilkan nilai testing dataset dengan akurasi yang lebih baik daripada konfigurasi yang lain.

4.5.2 Testing Dataset

Pengujian dilakukan menggunakan sumber dataset gambar yang berasal dari *data open source* [github](#). Pengujian dilakukan dalam bentuk gambar yang di input ke dalam model. Pada pengujian dilakukan menggunakan empat skenario berdasarkan konfigurasi parameter model yang sudah dilakukan.

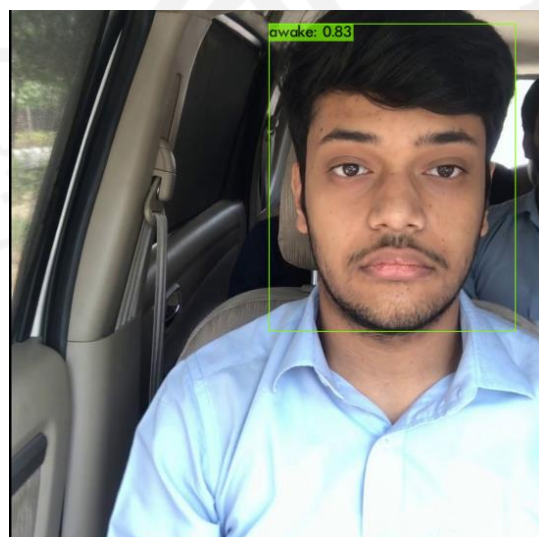
4.5.2.1 Skenario 1 (Split Dataset 90% : 10% ; Lr: 0.00261)

Hasil deteksi menunjukkan akurasi 94% mengantuk (*drowsy*) dengan kecepatan prediksi 15.67 ms seperti pada gambar 4.6. Nilai akurasi menunjukkan tingkat kepercayaan (*confidence*) sistem dalam melakukan deteksi.



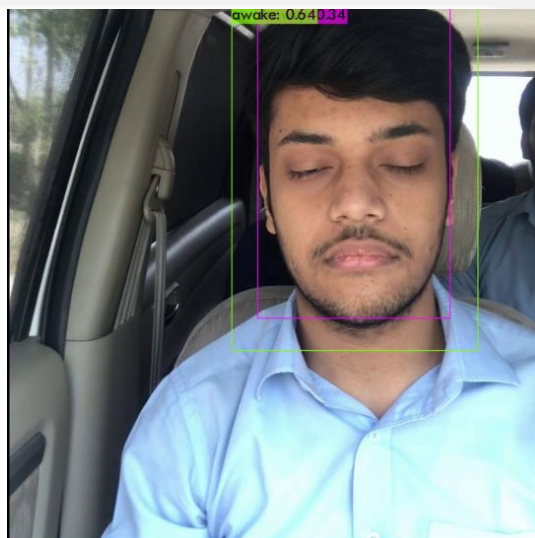
Gambar 4. 6. Testing pada gambar 1 skenario 1

Kemudian pada pengujian gambar yang kedua digunakan gambar dengan mata terbuka dan mulut tertutup yang menjadi salah satu ciri pengemudi yang terjaga (*awake*). Hasil deteksi menunjukkan akurasi 83% terjaga (*awake*) dengan kecepatan prediksi 15.48 ms seperti pada gambar 4.7.



Gambar 4. 7. Testing pada gambar 2 skenario 1

Pengujian selanjutnya dilakukan pada gambar dengan mata dan mulut tertutup yang menjadi ciri pengemudi yang mengantuk, namun sistem mendeteksi ganda yaitu 64% terjaga (*awake*) dan 34% mengantuk (*drowsy*) dengan kecepatan prediksi 15.52 ms. Bias deteksi dapat terjadi akibat sistem tidak mengenali pola gambar, hal ini terjadi karena pola gambar tidak dilatih sebelumnya sehingga muncul hasil deteksi ganda seperti pada gambar 4.8.



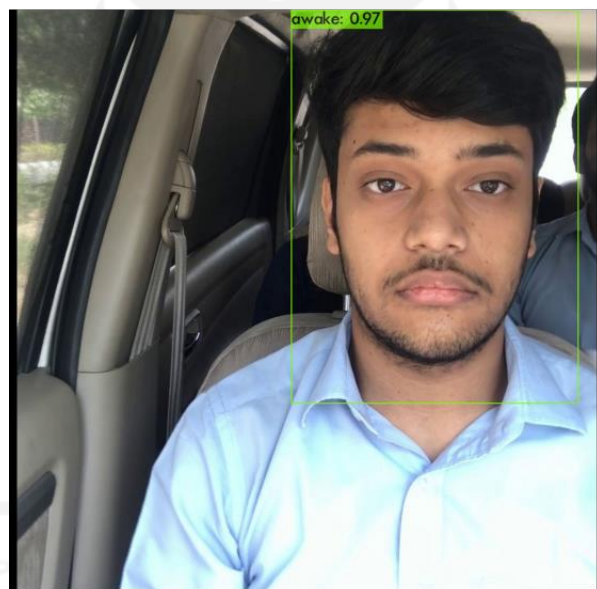
Gambar 4. 8. Testing pada gambar 3 skenario 1

4.5.2.2 Skenario 2 (Split Dataset 90% : 10% ; Lr: 0.001)

Hasil deteksi pada gambar 4.9 menunjukkan akurasi 100% mengantuk (*drowsy*), kemudian pada gambar 4.10 menunjukkan hasil deteksi 97% terjaga (*awake*). Sementara itu pada gambar 4.11 terdeteksi 96% terjaga (*awake*), hasil deteksi ini kurang sesuai karena seharusnya gambar terdeteksi mengantuk (*drowsy*). Hal ini dapat terjadi karena sistem mengenali gambar terdeteksi mengantuk (*drowsy*) apabila mata tertutup dan mulut terbuka, sehingga pada gambar dengan mata tertutup mulut tertutup terdeteksi sebagai pengemudi yang terjaga (*awake*).



Gambar 4. 9 Testing pada gambar 1 skenario 2



Gambar 4. 10 Testing pada gambar 2 skenario 2



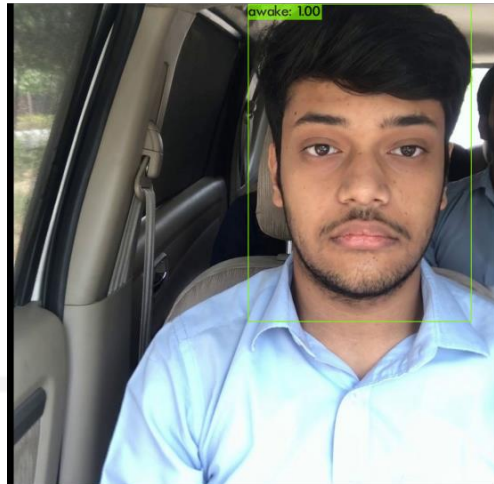
Gambar 4. 11. Testing pada gambar 3 skenario 2

4.5.2.3 Skenario 3 (Split Dataset 80% : 20% ; Lr: 0.00261)

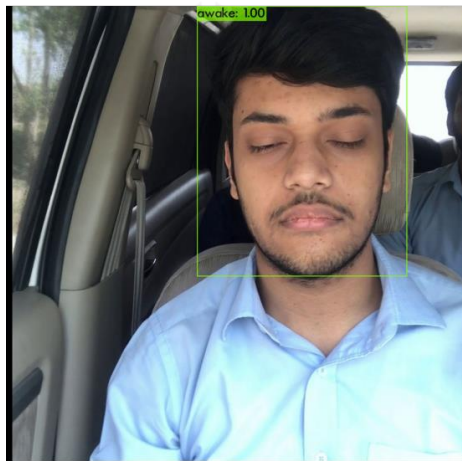
Skenario 3 merupakan skenario dengan konfigurasi dengan nilai IoU paling tinggi dan juga nilai loss paling kecil, yang akan berpengaruh pada akurasi sistem dalam melakukan deteksi. Pada gambar 4.12 menunjukkan hasil deteksi 100% pengemudi mengantuk (*drowsy*), kemudian pada gambar 4.13 menunjukkan hasil deteksi 100% pengemudi terjaga (*awake*). Namun pada gambar terakhir dengan wajah mengantuk, sistem mendeteksi 100% awake, hal ini terjadi karena meskipun mata tertutup tetapi mulut juga tertutup sehingga terdeteksi sebagai pengemudi yang terjaga (*awake*).



Gambar 4. 12. Testing pada gambar 1 skenario 3



Gambar 4. 13. Testing pada gambar 2 skenario 3



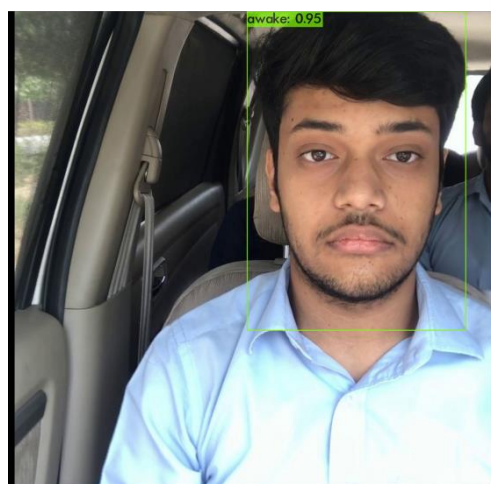
Gambar 4. 14. Testing pada gambar 3 skenario 3

4.5.2.4 Skenario 4 (Split Dataset 80% : 20% ; Lr: 0.001)

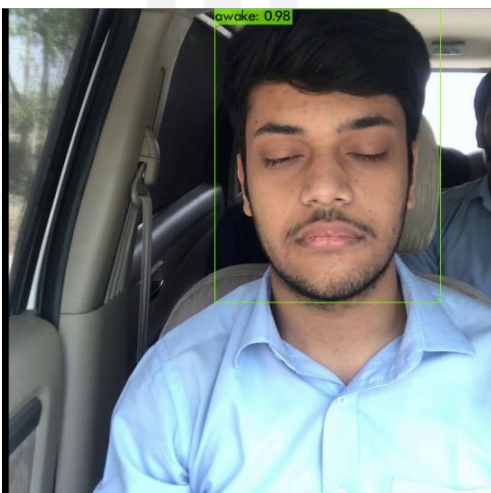
Pengujian pada skenario keempat pada gambar 4.15 menunjukkan hasil deteksi 98% pengemudi mengantuk (*drowsy*), kemudian pada gambar 4.16 menunjukkan hasil deteksi 95% pengemudi terjaga (*awake*). Namun pada gambar terakhir, 4.17, sama halnya dengan skenario 2 dan 3 sistem mendeteksi 100% awake, hal ini terjadi karena meskipun mata tertutup tetapi mulut juga tertutup sehingga terdeteksi sebagai pengemudi yang terjaga (*awake*).



Gambar 4. 15. Testing pada gambar 1 skenario 4



Gambar 4. 16. Testing pada gambar 2 skenario 4



Gambar 4. 17. Testing pada gambar 3 skenario 4

BAB V

ANALISIS DAN PEMBAHASAN

Berdasarkan sistem deteksi kantuk pada pengemudi yang sudah dirancang, diperoleh hasil algoritma deteksi objek *you only look once* (YOLO) dapat diimplementasikan untuk melakukan deteksi pada pengemudi dengan label mengantuk (*drowsy*) dan terjaga (*awake*) dengan nilai rata-rata IoU terbesar 84.62% dengan split dataset sebesar 80% training dan 20% testing. Nilai IoU tersebut menunjukkan tingkat kepercayaan model dapat melakukan evaluasi pada area objek deteksi atau *bounding box*. IoU bukanlah tingkat kepercayaan dalam memprediksi apakah pengemudi sedang dalam keadaan *drowsy* atau *awake*, IoU hanya menunjukkan akurasi dari *bounding box* yang sudah dibuat karena tidak berkaitan dengan label data yang sudah ditentukan.

Pengumpulan dataset dan proses pelabelan menjadi hal yang krusial dalam membuat rancangan sistem deteksi menggunakan YOLO. Semakin banyak jumlah dataset yang dilatih untuk merancang sistem deteksi semakin akurat hasil deteksi yang didapatkan. Hal ini terjadi karena semakin banyak data yang dilatih maka sistem akan semakin banyak mengenali ragam ekspresi dan posisi gambar saat dilakukan pengujian (*testing*) sehingga dapat memberikan hasil deteksi dengan akurasi yang tinggi.

Ada beberapa hal yang menyebabkan nilai akurasi pada penelitian ini tidak terlalu tinggi dan juga kekeliruan dalam melakukan prediksi. Dataset yang digunakan saat proses pelatihan pada penelitian ini menggunakan gambar yang general yaitu hanya menggunakan gambar wajah untuk area deteksi, tidak menggunakan dataset dengan area yang lebih spesifik seperti dataset mata tertutup dan dataset mulut yang terbuka sebagai indikasi pengemudi sedang mengantuk, hal ini menyebabkan keterbatasan sistem dalam mengenali ragam ekspresi pengemudi. Pada penelitian ini hanya menggunakan 200 dataset gambar pengemudi yang terjaga (*awake*) dan mengantuk (*drowsy*), hal ini terjadi karena dataset untuk pengemudi dengan wajah mengantuk tidak banyak tersedia pada data *open source*. Dalam keadaan mengemudi sulit untuk mengambil gambar pengemudi

saat mengantuk karena pada praktiknya saat pengemudi mengantuk sebaiknya untuk mengambil jeda dan tidak melanjutkan perjalanan terlebih dahulu.

Hal lain yang dapat menjadi penyebab akurasi yang tidak tinggi adalah terjadinya *human error* saat melakukan proses *labelling* karena dilakukan secara manual. Contoh *human error* yang dimaksud terjadi ketika membuat *bounding box* untuk setiap gambar, ada *bounding box* yang dibuat terlalu lebar melebihi bentuk wajah dan terdapat *bounding box* yang dibuat terlalu pas sehingga tidak semua wajah terlabeli dengan baik. Pada penelitian yang dilakukan oleh (Sager et al., 2021) yang berjudul “*A survey of image labelling for computer vision applications*” disebutkan jika proses *labelling* menghabiskan banyak waktu karena dilakukan secara manual, hal ini disebabkan hingga penelitiannya dilakukan belum ada penelitian sistematis yang membahas tentang *image labelling software* (ILS) untuk penelitian *computer vision*. Oleh karena itu pada penelitiannya kali ini diberikan rekomendasi ILS yang dapat digunakan diantaranya *LabelMe*, *labelImg*, *The Visual Geometry Group Image Annotator* (VIA), *Intel’s CV Annotation Tool* (CVAT), *Microsoft’s Visual Object Tagging Tool* (VoTT), *Web Annotation*, dan *ImageTagger*.

Selain itu, proses *data split* atau pembagian berapa banyak jumlah data yang akan digunakan untuk melakukan pelatihan (*training*) dan pengujian (*testing*) juga tidak kalah penting dalam merancang sistem deteksi karena pada saat pelatihan kita mengajari sistem untuk mempelajari pola dari data yang sudah diberikan, pada penelitian ini berarti sistem dilatih untuk mengenali wajah pengemudi yang mengantuk dan terjaga dari gambar-gambar yang sudah diberikan label sebelumnya seperti pada poin 4.2.

Selanjutnya pada tahap konfigurasi parameter model dilakukan penyesuaian beberapa parameter yaitu mengganti jumlah kelas yang digunakan dan juga berapa kali iterasi dilakukan. Konfigurasi ini dilakukan agar mendapatkan hasil yang optimal dalam melakukan deteksi dan diperoleh akurasi yang sesuai. Pada penelitian ini dilakukan 4000 iterasi yang menyesuaikan dengan jumlah kelas penelitian. Nilai 4000 ini diperoleh dari hasil perhitungan *max batch* yang merupakan batas iterasi dalam melakukan training yang sudah dituliskan pada poin 4.4.4 sebelumnya.

Harapannya dengan adanya sistem deteksi kantuk pada pengemudi dapat diimplementasikan secara lebih lanjut pada sistem peringatan yang berupa alarm yang nantinya dapat langsung diterapkan saat mengemudi. Dengan adanya alarm deteksi kantuk pada pengemudi harapannya dapat membantu meminimalisir terjadinya

kecelakaan yang dapat membahayakan pengemudi maupun penumpang yang berada di dalamnya. Penelitian ini masih memiliki banyak kekurangan dalam melakukan deteksi seperti dijelaskan pada paragraf sebelumnya dan juga sistem belum dapat melakukan deteksi secara *real time*.



BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan dari hasil pengolahan data dan hasil pembahasan pada penelitian ini, dapat ditunjukkan bahwa arsitektur *convolutional neural network* (CNN) dengan algoritma *you only look once* (YOLO) dapat diimplementasikan untuk melakukan deteksi pada pengemudi apakah mengantuk (*drowsy*) atau terjaga (*awake*). Selain itu dengan melakukan konfigurasi parameter model dengan *batch size* 64, *network size* 416x416, *subdivisions* 16, *max batch* 4000, *filters* 21, dengan empat skenario training data dan testing data dengan learning rate 0.00261 dan 0.001 dengan split dataset 90%:10% dan 80%:10% diperoleh nilai IoU terbesar pada konfigurasi menggunakan split dataset 80%:20% dengan learning rate 0.00261.

6.2 Saran

Berdasarkan penelitian ini, saran yang dapat diberikan pada penelitian selanjutnya adalah:

1. Menambahkan jumlah dataset dengan kategori yang lebih spesifik pada proses training data agar model dapat mengenali pola gambar lebih baik dan optimal.
2. Melakukan training data pada komputer dengan spesifikasi GPU yang lebih tinggi agar proses training data dapat berlangsung lebih cepat.
3. Melakukan konfigurasi parameter model lebih baik agar diperoleh hasil deteksi yang optimal dan dapat melakukan sistem deteksi kantuk secara *real time*.

DAFTAR PUSTAKA

- Al, T., Hadi, A., Usman, K., Saidah, S., Telekomunikasi, T., Elektro, F. T., & Telkom, U. (2021). *People Counting for Public Transportations Using You Only Look Once Method People Counting Untuk Transportasi Publik Menggunakan*. 2(1), 57–66.
- Almogbel, M. A., Dang, A. H., & Kameyama, W. (2018). EEG-signals based cognitive workload detection of vehicle driver using deep learning. *International Conference on Advanced Communication Technology, ICACT, 2018-February*, 256–259.
- Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Van Esesn, B. C., Awwal, A. A. S., & Asari, V. K. (2018). *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*.
- Bi, X., Chen, Z., & Yue, J. (2020). A Novel One-step Method Based on YOLOv3-tiny for Fatigue Driving Detection. *2020 IEEE 6th International Conference on Computer and Communications, ICC 2020*, 1241–1245.
- Biju, A., & Edison, A. (2020). Drowsy Driver Detection Using Two Stage Convolutional Neural Networks. *2020 IEEE Recent Advances in Intelligent Computational Systems, RAICS 2020*, 7–12.
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*.
- Dhiaegana, R. N. (2020). Penerapan Convolutional Neural Network untuk Deteksi Pedestrian pada Sistem Autonomous Vehicle. *Institut Teknologi Bandung*.
- Faraji, F., Lotfi, F., Khorramdel, J., Najafi, A., & Ghaffari, A. (2021). *Drowsiness Detection Based On Driver Temporal Behavior Using a New Developed Dataset*. 1–7.
- Haenlein, M., & Kaplan, A. (2019). A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, 61(4), 5–14.
- Hemdan, E. E. (n.d.). *COVIDX-Net: A Framework of Deep Learning Classifiers to Diagnose COVID-19 in X-Ray Images*.
- Jessar, H. F., Wibowo, A. T., & Rachmawati, E. (2021). Klasifikasi Genus Tanaman Sukulen Menggunakan Convolutional Neural Network. *E-Proceeding of Engineering*, 8(2), 3180–3196.
- Khairunnas, Yuniarno, E. M., & Zaini, A. (2021). *Pembuatan Modul Deteksi Objek Manusia Menggunakan Metode YOLO untuk Mobile Robot* (Vol. 10, Issue 1).
- Liu, C., Tao, Y., Liang, J., Li, K., & Chen, Y. (2018). Object detection based on YOLO network. *Proceedings of 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference, ITOEC 2018, Itoec*, 799–803.
- Mahdi, F. P., Motoki, K., & Kobashi, S. (2020). Optimization technique combined with deep learning method for teeth recognition in dental panoramic radiographs.

- Scientific Reports*, 10(1), 1–12.
- Ngxande, M., Tapamo, J.-R., & Burke, M. (2017). *Driver drowsiness detection using Behavioral measures and machine learning techniques: A review of state-of-art techniques*. 156–161.
- Pan, J. (2020). The drowsy testing system according to deep-learning. *Proceedings - 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering, AEMCSE 2020, 1*, 247–251.
- Prabowo, D. A., & Abdullah, D. (2018). Deteksi dan Perhitungan Objek Berdasarkan Warna Menggunakan Color Object Tracking. *Pseudocode*, 5(2), 85–91.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 779–788.
- Sager, C., Janiesch, C., & Zschech, P. (2021). A survey of image labelling for computer vision applications. *Journal of Business Analytics*, 4(2), 91–110.
- Saha, S. (2018). A comprehensive guide to convolutional neural networks—the ELI5 way. *Towards Data Science* 15.
- Sinha, A., Aneesh, R. P., & Gopal, S. K. (2021). Drowsiness Detection System Using Deep Learning. *Proceedings of 2021 IEEE 7th International Conference on Bio Signals, Images and Instrumentation, ICBSII 2021*, 1–6.
- Suyanto, Ramadhani, K. N., & Mandala, S. (2019). *Deep Learning Modernisasi Machine Learning untuk Big Data* (1st ed.). INFORMATIKA.
- Tao, J., Wang, H., Zhang, X., Li, X., & Yang, H. (2018). An object detection system based on YOLO in traffic scene. *Proceedings of 2017 6th International Conference on Computer Science and Network Technology, ICCSNT 2017, 2018-Janua*, 315–319.
- Tjolleng, A., Jung, K., Hong, W., Lee, W., Lee, B., You, H., Son, J., & Park, S. (2017). Classification of a Driver's cognitive workload levels using artificial neural network on ECG signals. *Applied Ergonomics*, 59, 326–332.
- Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018.

LAMPIRAN

Lampiran *script cloning* framework darknet

```
!git clone https://github.com/AlexeyAB/darknet

Cloning into 'darknet'...
remote: Enumerating objects: 15313, done.
remote: Total 15313 (delta 0), reused 0 (delta 0), pack-reused 15313
Receiving objects: 100% (15313/15313), 13.71 MiB | 17.97 MiB/s, done.
Resolving deltas: 100% (10405/10405), done.
```

Lampiran *script file .names*

```
drowsy
awake
```

Lampiran *script file .data*

```
classes = 2
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /mydrive/drowsy_detection/training
```

Lampiran *script split dataset*

```
import glob, os

# Current directory
current_dir = os.path.dirname(os.path.abspath(__file__))

print(current_dir)

current_dir = 'data/obj'

# Percentage of images to be used for the test set
percentage_test = 10;

# Create and/or truncate train.txt and test.txt
file_train = open('data/train.txt', 'w')
file_test = open('data/test.txt', 'w')

# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndFilename in glob.iglob(os.path.join(current_dir, '*.jpg')):
    title, ext = os.path.splitext(os.path.basename(pathAndFilename))

    if counter == index_test:
        counter = 1
        file_test.write("data/obj" + "/" + title + '.jpg' + "\n")
    else:
        file_train.write("data/obj" + "/" + title + '.jpg' + "\n")
```

```
counter = counter + 1
```

Lampiran *script* mount *google drive*

```
#mount drive
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')

# this creates a symbolic link so that now the path /content/gdrive/My\ Drive/ is equal to /mydrive
!ln -s /content/gdrive/My\ Drive/ /mydrive

# list contents in yolov4-tiny folder in your drive
!ls /mydrive/drowsy_detection
```

Lampiran *script* aktivasi GPU dan OPENCV

```
# change makefile to have GPU and OPENCV enabled
# also set CUDNN, CUDNN_HALF and LIBSO to 1

%cd /content/darknet/
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

Lampiran *script* menjalankan darknet

```
# build darknet
!make
```

Lampiran *script* Insert file yang dibutuhkan pada darknet

```
# Clean the data and cfg folders first except the labels folder in data which is required

%cd data/
!find -maxdepth 1 -type f -exec rm -rf {} \;
%cd ..

%rm -rf cfg/
%mkdir cfg
```

```
#copy the datasets zip file to the root darknet folder
!cp /mydrive/drowsy_detection/obj.zip ../
```

```
# unzip the datasets and their contents so that they are now in /darknet/data/ folder
!unzip ../obj.zip -d data/
```

```
#copy the custom cfg file from the drive to the darknet/cfg folder
!cp /mydrive/drowsy_detection/yolov4-tiny-custom.cfg ./cfg
```

```
# copy the obj.names and obj.data files so that they are now in /darknet/data/ folder
!cp /mydrive/drowsy_detection/obj.names ./data
!cp /mydrive/drowsy_detection/obj.data ./data
```

```
#copy the process.py file from the drive to the darknet directory
!cp /mydrive/drowsy_detection/process.py ./
```

Lampiran *script* menjalankan data training

```
# run process.py ( this creates the train.txt and test.txt files in our darknet/data folder )
!python process.py

# list the contents of data folder to check if the train.txt and test.txt files have been created
!ls data/
```

Lampiran *script* mendownload file *pre-trained yolov4-weights*

```
# Download the yolov4-tiny pre-trained weights file
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.conv.29
```

Lampiran *script* melakukan training data

```
# train your custom detector! (uncomment %%capture below if you run into memory issues or your Colab is crashing)
# %%capture

!./darknet detector train data/obj.data cfg/yolov4-tiny-custom.cfg yolov4-tiny.conv.29 -dont_show -map
```

Lampiran *script* mengecek performa sistem yang dibuat

```
#You can check the mAP for all the saved weights to see which gives the best results ( xxxx here is the saved weight number like 4000, 5000 or 6000 and so on )

!./darknet detector map data/obj.data cfg/yolov4-tiny-custom.cfg /mydrive/drowsy_detection/training/yolov4-tiny-custom_best.weights -points 0
```

Lampiran *script* melakukan deteksi pada gambar dan video

```
#set your custom cfg to test mode
%cd cfg
!sed -i 's/batch=64/batch=1/' yolov4-tiny-custom.cfg
!sed -i 's/subdivisions=16/subdivisions=1/' yolov4-tiny-custom.cfg
%cd ..
```

```
# run your custom detector with this command (upload an image to your google drive to test, the thresh flag sets the minimum accuracy required for object detection)

!./darknet detector test data/obj.data cfg/yolov4-tiny-custom.cfg /mydrive/drowsy_detection/training/yolov4-tiny-custom_best.weights /content/gdrive/MyDrive/drowsy_detection/images/6.jpg -thresh 0.3
imshow('predictions.jpg')
```

Lampiran script yolov4-tiny-custom.cfg

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.00261
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
```

```
pad=1
activation=leaky

[route]
layers=-1
groups=2
group_id=1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[route]
layers = -1,-2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -6,-1

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[route]
layers=-1
groups=2
group_id=1

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
```

```

size=3
stride=1
pad=1
activation=leaky

[route]
layers = -1,-2

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -6,-1

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[route]
layers=-1
groups=2
group_id=1

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[route]
layers = -1,-2

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -6,-1

```

```

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

#####

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=21
activation=linear

[yolo]
mask = 3,4,5
anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319
classes=2
num=6
jitter=.3
scale_x_y = 1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh = .7
truth_thresh = 1
random=0
resize=1.5
nms_kind=greedynms
beta_nms=0.6

[route]
layers = -4

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

```



```
[upsample]
stride=2

[route]
layers = -1, 23

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=21
activation=linear

[yolo]
mask = 0,1,2
anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319
classes=2
num=6
jitter=.3
scale_x_y = 1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh = .7
truth_thresh = 1
random=0
resize=1.5
nms_kind=greedynms
beta_nms=0.6
```