Pengembangan Model Klasifikasi Mata Tertutup dan Terbuka Dalam Identifikasi Kelelahan Menggunakan Arsitektur Mobile CNN

Maghfirah Suyuti
18917213

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer
Konsentrasi Magister Informatika
Program Studi Informatika Program Magister
Fakultas Teknologi Industri
Universitas Islam Indonesia
2023

Lembar Pengesahan Pembimbing

Pengembangan Model Klasifikasi Mata Tertutup dan Terbuka Dalam Identifikasi Kelelahan Menggunakan Arsitektur Mobile CNN

Maghfirah Suyuti
18917213

Yogyakarta, 26 Januari 2023

Pembimbing

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.

Lembar Pengesahan Penguji

Pengembangan Model Klasifikasi Mata Tertutup dan Terbuka Dalam Identifikasi Kelelahan Menggunakan Arsitektur Mobile CNN

Maghfirah Suyuti
18917213

Yogyakarta, 26 Januari 2023

Tim Penguji,

Dhomas Hatta Fudholi, ST., M.Eng, Ph.D
Ketua

Dr. Syarif Hidayat, ST., MIT
Anggota I

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D
Anggota II

Mengetahui,
Ketua Program Studi Informatika Program Magister
Universitas Islam Indonesia

Irving Vitra Paputungan, ST., M.Sc. Ph.D

Abstrak

Pengembangan Model Klasifikasi Mata Tertutup dan Terbuka Dalam Identifikasi Kelelahan Menggunakan Arsitektur Mobile CNN

Dampak *lockdown* di masa pandemi di Indonesia yang dimulai tahun 2020 mempengaruhi adaptasi pelajar beraktivitas di rumah dan harus berada didepan layar selama proses belajar tiap hari. Hal tersebut bisa membuat pelajar kelelahan terutama mempercepat kelelahan mata dan mengalami stress, sehingga mempengaruhi tingkat keberhasilan akademik. Penelitian ini dilakukan klasifikasi kelelahan berdasarkan fitur mata untuk mengetahui indikasi kelelahan yang dialami pelajar. Pada penelitian ini digunakan deep learning dengan membandingkan *mobile architecture* Convolution Neural Network (CNN). Dataset yang digunakan adalah MRL Eye Dataset. Dataset ini telah memiliki anotasi *eye state*. Anotasi *eye state* digunakan untuk klasifikasi kondisi mata dengan 2 label, yaitu label 0 untuk anotasi mata tertutup dan label 1 untuk anotasi mata terbuka. Proses training yang dilakukan menggunakan data yang berukuran 224x224 dan pengujian dengan jumlah 50 epoch dan 0.0001 learning rate. Untuk memberikan analisis yang lebih luas dan untuk mendapatkan *best model,* penulis membandingkan dan mengevaluasi model training terhadap tiga arsitektur yakni MobilenetV1, MobileNetV3 dan EfficientNet Lite0. Penentuan *best model* dilakukan dengan melihat ukuran model terkecil dan performa model. Meskipun, hasil performa metrik EfficientNet Lite0 didapatkan presisi sebesar 1.00, recall 1.00, f1-score 1.00 dan MobileNetV3 Large lebih unggul dalam nilai prediksi dengan nilai *false negative* 0 dan CP 32. MobileNetV1 menghasilkan *best model* dengan ukuran model terkecil sesuai dengan tujuan penelitian yaitu 12 MB dibandingkan MobileNetV3 Large dan EfficientNet Lite0, akan tetapi tidak mengurangi akurasi prediksi sebesar 98% dengan inference time 0.05 detik. Selanjutnya, pengujian model pada sistem pendeteksi kelelahan yang dikembangkan oleh Ashish Kushwaha berhasil membaca kondisi mata subjek melalui *webcam* komputer. Subjek terdeteksi lelah karena mata tertutup dengan nilai *score* 21 atau selama 8 detik.

Kata kunci

eye drowsiness, cnn, tensorflow, mobilenet, efficientnetlite

Abstract

Model Development of Closed and Open Eye Classification in Fatigue Identification Using CNN Mobile Architecture

The impact of the lockdown during the pandemic in Indonesia which began in 2020 affected the adaptation of students to activities at home and having to be in front of a screen during the learning process every day. This can make students tired, especially accelerate eye fatigue and experience stress, thus affecting the level of academic success. In this study, classification of fatigue was carried out based on eye features to find out indications of fatigue experienced by students. In this study, deep learning is used by comparing the mobile architecture Convolution Neural Network (CNN). The dataset used is the MRL Eye Dataset. This dataset already has annotation eye status. The eye state annotation is used to classify eye conditions with 2 labels, namely label 0 for closed eye annotation and label 1 for open eye annotation. The training process is carried out using data measuring 224x224 and testing with a total of 50 epochs and 0.0001 learning rate. In order to provide a broader analysis and obtain the best model, the authors compare and evaluate model training for three architectures namely MobilenetV1, MobileNetV3 and EfficientNet Lite0. The best model maintenance is done by looking at the smallest model size and performance model. Although, the superior performance results of the EfficientNet Lite0 metric get a precision of 1.00, recall of 1.00, f1-score of 1.00 and MobileNetV3 Large are more predictive with false negative values of 0 and CP 32. MobileNetV1 output has the best model with the smallest model size of 12 MB as our main goal, compared to MobileNetV3 Large of 32 MB and EfficientNet Lite0 of 24 MB, but does not reduce the performance of the model with a prediction accuracy of 98% and an inference time of 0.05 seconds. Furthermore, the test model for the fatigue detection system developed by Ashish Kushwaha managed to read the subject's eye condition via a computer webcam. Subjects were detected as tired because their eyes were closed with a score of 21 or for 8 seconds.

Keywords

eye drowsiness, cnn, tensorflow, mobilenet, efficientnetlite

Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

akarta, 26 Januari 2023

Maghfirah Suyuti, S.Kom

Daftar Publikasi

<u>Publikasi yang menjadi bagian dari tesis</u>

Fudholi, D. H., Nayoan, R. A. N., Suyuti, M., & Rahmadi, R. (2021). Deteksi Indikasi Kelelahan Menggunakan Deep Learning. J-SAKTI (Jurnal Sains Komputer dan Informatika), 5(1), 1-9.

Sitasi publikasi 1

Kontributor	Jenis Kontribusi
Dhomas Hatta Fudholi	Mendesain eksperimen Menulis dan merevisi *paper*
Royan Abida Nur Nayoan	Membuat model dan menulis *paper*
Maghfirah Suyuti	Membuat model dan menulis *paper*
Ridho Rahmadi	Mendesain eksperimen

Halaman Kontribusi

Dalam penulisan tesis ini pembimbing memberikan beberapa masukkan sebagai perbaikan dari cara penulisan tesis serta memberikan saran tentang data yang akan diolah, dan dianalisis. Selain itu, pembimbing juga selalu memberikan dukungan terhadap proses penelitian yang awalnya penulis rasa sangat sulit untuk dilakukan hingga akhirnya menemukan jalan yang lebih mudah.

Halaman Persembahan

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Kupersembahkan Tesis ku ini dengan setulus hati untuk :

Orang tuaku tercinta (Ibuku Ramlah dan Ayahku Suyuti Yusuf) yang selalu mengiringi langkah kakiku dengan do'a, yang tak pernah putus asa untuk mengurus, mendidik, menasehati, menyemangati, membimbing serta memberikan motivasi kepadaku sehingga bisa terus lanjut menyelesaikan tesis ini.

Kata Pengantar

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Alhamdulillahirabbil'alamin, segala puji syukur kehadirat Allah SWT senantiasa penulis panjatkan karena atas limpahan rahmat dan hidayah-Nya, tesis yang berjudul "Klasifikasi Kelelahan Berdasarkan Fitur Kondisi Mata Menggunakan Metode CNN" dapat berjalan dengan lancar dalam penyelesaiannya. Tesis ini diajukan sebagai bagian dalam menyelesaikan studi dan sebagai salah satu syarat untuk memperoleh gelar Magister Komputer pada Program Studi Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Dalam penyelesaian Tesis ini, penulis banyak mendapatkan bantuan dari berbagai pihak, untuk itu penulis menyampaikan ucapan terima kasih setulusnya kepada:

1. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
2. Bapak Irving Vitra Paputungan, Ph.D., selaku Ketua Program Studi Teknik Informatika Program Studi Magister Universitas Islam Indonesia.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku dosen pembimbing yang sejak awal banyak membantu penulis dalam memberikan ide, saran dan kritiknya.
4. Bapak "penguji", selaku dewan penguji tesis yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
5. Dosen Program Studi Magister Teknik Informatika yang telah memberikan bekal ilmu pengetahuan kepada penulis, semoga ilmunya menjadi amal jariyah di dunia maupun akhirat.
6. Staff Akademik Program Pascasarjana Fakultas Teknologi Universitas Islam Indonesia, yang telah membantu dalam segala urusan administrasi di kampus.
7. Sahabat Sains Data Angkatan 2 yang telah turut serta membantu, mendukung, menyemangati, terkhusus kepada mas Miftah dan Eko karena telah memberikan masukan dan meluangkan waktu kepada penulis untuk mengajar dan menjawab semua pertanyaan-pertanyaan saya selama menjalani penelitian.

Penulis menyadari bahwa dalam penulisan tesis ini masih banyak kelemahan dan kekurangan, untuk itu kritik dan saran yang sifatnya membangun sangat penulis harapkan agar tesis ini dapat menjadi lebih baik.

Daftar Isi

Lembar Pengesahan Pembimbing ii

Lembar Pengesahan Penguji iii

Abstrak iv

Abstract v

Pernyataan Keaslian Tulisan vi

Daftar Publikasi vii

Halaman Kontribusi viii

Halaman Persembahan ix

Kata Pengantar x

Daftar Isi xi

Daftar Tabel xiii

Daftar Gambar xiv

Glosarium xvi

BAB 1 Pendahuluan 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 3

1.3 Tujuan Penelitian 3

1.4 Batasan Masalah 3

1.5 Manfaat Penelitian 3

1.6 Sistematika Penulisan 3

BAB 2 Tinjauan Pustaka 6

2.1 Penelitian Terdahulu 6

2.2 Konsep Pengetahuan Model yang Digunakan 8

2.2.1 MobileNet 8

2.2.2 EfficientNet Lite 11
2.2.3 Kelelahan Mata 12
BAB 3 Metodologi 14
3.1 Tahapan Penelitian 14
3.1.1 Data Gathering 14
3.1.2 Data Training 16
3.1.3 Model Evaluation 18
3.2 Analisis Kebutuhan 19
3.2.1 Analisis Kebutuhan Masukan 19
3.2.2 Analisis Kebutuhan Perangkat Lunak 19
3.2.3 Analisis Kebutuhan Keluaran 19
BAB 4 Hasil dan Pembahasan 20
4.1 Dataset yang digunakan 20
4.2 Tahapan training 21
4.3 Model Evaluation 26
4.4 Impelementasi dan pengujian model pada sistem deteksi kelelahan 28
4.5 Source Code Training 30
4.6 Source Code Confusion Matrix 37
BAB 5 Kesimpulan dan Saran 41
5.1 Kesimpulan 41
5.2 Saran 41
Daftar Pustaka 42
LAMPIRAN A 45

Daftar Tabel

Tabel 2.1 *State of the Art* penelitian 7
Tabel 3.1 Keterangan Anotasi 16
Tabel 4.1 Perbandingan *accuracy* dan *loss* model 25
Tabel 4.2 Perbandingan Precision, Recall, F1-Score 26
Tabel 4.3 Hasil Perbandingan TP, FP, TN, FN, CP dan I 27
Tabel 4.4 Perbandingan *inference time* 28

Daftar Gambar

Gambar 1.1 Gambaran sistem pendeteksi kelelahan yang digunakan 2
Gambar 2.1 Pembagian konvolusi MobileNet 9
Gambar 2.2 Konvolusi MobileNet 9
Gambar 2.3 Arsitektur MobileNetV1 10
Gambar 2.4 Susunan *hidden layers* MobileNetV3 10
Gambar 2.5 Arsitektur MobileNetV3 Large 11
Gambar 2.6 Hasil ukuran model (mb) dan akurasi EfficientNet-Lite 12
Gambar 2.7 Hasil ukuran model (mb) dan akurasi EfficientNet-Lite 12
Gambar 3.1 Tahapan penelitian 14
Gambar 3.2 Contoh dataset MRL Eye Dataset 15
Gambar 3.3 Struktur folder *prepared data* 16
Gambar 3.4 Proses *Convolusional Neural network* 17
Gambar 3.5 Alur proses training 17
Gambar 3.6 Tahapan klasifikasi MobileNetv1, Mobilenetv3 dan Efficientnet lite0 18
Gambar 4.1 *Split* dataset berdasarkan anotasi *eye state* 20
Gambar 4.2 Label *open* dan *closed* pada gambar 21
Gambar 4.3 Hasil augmentasi gambar mata 22
Gambar 4.4 Hasil plot *accuracy* dan *cross entrophy* MobileNetV1 23
Gambar 4.5 Hasil plot *accuracy* dan *cross entrophy* MobileNetV3 Large 24
Gambar 4.6 Hasil plot *accuracy* dan *cross entrophy* EfficientNet Lite0 25
Gambar 4.7. *Heatmap* hasil prediksi 27
Gambar 4.8 Deteksi mata lelah dari perubahan *score prediction* 29
Gambar 4.9 Mata terdeteksi terbuka 29
Gambar 4.10 Mata terdeteksi tertutup 30
Gambar 4.11 *Library* yang digunakan dan cek status GPU perangkat komputasi 31
Gambar 4.12 Pengaturan *storage* yang digunakan 31
Gambar 4.13 Pengaturan data sebelum pelatihan 32
Gambar 4.14 Plot label gambar *open* dan *closed* 32
Gambar 4.15 Pembagian data 32
Gambar 4.16 Penggunaan *buffered prefetching* 33
Gambar 4.17 Teknik augmentasi gambar 33
Gambar 4.18 Pengaturan *base model* 34

Gambar 4.19 Pembuatan model dengan Keras 34
Gambar 4.20 *Compile* model 34
Gambar 4.21 *Output model* 35
Gambar 4.22 *Training* dengan 25 epoch 35
Gambar 4.23 *Un-freeze base model* 35
Gambar 4.24 Penggunaan *fine tuning* 36
Gambar 4.25 Model di *compile* kembali 36
Gambar 4.26 *Training* dengan 25 epoch tambahan *fine tuning* 36
Gambar 4.27 *Save model* 36
Gambar 4.28 *Import library* 37
Gambar 4.29 Define data gambar dan *output logit* 37
Gambar 4.30 Pencarian jumlah TP, FP, TN dan FN 38
Gambar 4.31 Pencarian jumlah CP dan ICP 39
Gambar 4.32 Evaluasi *confusion matrix* 39
Gambar 4.33 *Metrics performance* 40
Gambar 4.34 Menampilkan plot *heatmap* 40

Glosarium

UNICEF	- UNITED NATIONS CHILDRENS FUND
FLOP	- Floating Points Performance Per Second
TF	- TensorFlow
CNN	- Convolutional Neural Network
CPU	- Central Processing Unit
GPU	- Graphic Processing Unit

BAB 1
Pendahuluan

1.1 Latar Belakang

Sejak pandemi di awal Maret 2020 hingga saat ini, proses belajar mengajar saat ini lebih banyak dilakukan secara daring meskipun beberapa instansi sudah mulai melakukan tatap muka atau secara luring. Pada sistem pembelajaran daring, pelajar diharapkan berada di depan komputer dan memperhatikan guru atau dosen. Pelajar harus mengikuti proses belajar dari jarak jauh seperti di rumah masing-masing dengan berbagai macam kondisi. Berdasarkan survei yang dilakukan UNICEF pada 18-29 Mei 2020 dan 5-8 Juni 2020, sebanyak 66 persen dari 60 juta siswa dari berbagai jenjang pendidikan di 34 provinsi mengaku tidak nyaman belajar dirumah (UNICEF, 2019). Selain itu, berada di depan komputer selama berjam-jam lebih cepat mengalami kelelahan mata.

Pada penelitian ini, dilakukan klasifikasi kondisi mata untuk mengetahui pelajar yang mengalami indikasi kelelahan saat berada di depan komputer. Indikasi kelelahan dapat dilihat pada ekspresi wajah (Fudholi dkk., 2021). Namun, secara fitur mata pada wajah akan digunakan untuk mengetahui kelelahan. Hal ini dikarenakan, ketika seseorang berusaha menahan rasa lelahnya dan meskipun wajahnya masih terlihat baik, kondisi mata akan lebih mudah di identifikasi untuk mengetahui apakah seseorang telah mengalami kelelahan atau tidak. Terkadang ketika kita kelelahan, mata cenderung sering tertutup dengan waktu yang relatif lebih lama dan berkedip secara berkala untuk mengatur fokus penglihatan. Sehingga, identifikasi kelelahan dapat dilakukan dengan pengembangan model klasifikasi kondisi mata berdasarkan pergerakan mata terbuka dan tertutup.

Pengembangan model klasifikasi kondisi mata tertutup dan terbuka dapat dilakukan dengan mengumpulkan dataset mata dengan beragam kondisi. Penelitian ini menggunakan dataset MRL Eye dan akan fokus pada pengembangan model klasifikasi dengan memori kecil yang dapat digunakan pada *resource* terbatas. Klasifikasi kelelahan berdasarkan fitur mata akan dilakukan menggunakan *mobile architecture* CNN. MobileNet adalah salah satu arsitektur convolutional neural network (CNN) berbasis ponsel yang dapat digunakan untuk mengatasi kebutuhan akan computing resource berlebih (Howard dkk., 2019), sehingga cocok untuk diterapkan pada perangkat dengan memori dan CPU terbatas.

MRL Eye Dataset juga digunakan pada penelitian (Suresh dkk., 2021) dan mengevaluasi 48.000 gambar menggunakan model CNN, menunjukkan akurasi 86,05%. Dalam penelitian ini, deep learning digunakan untuk menyarankan sebuah frame baru yang mengklasifikasikan kondisi mata pengemudi, yaitu terbuka atau tertutup. Ketika seorang pengemudi dideteksi mengantuk, sistem akan membunyikan bip.

Penelitian lain (Adianto, 2021) melakukan pendekatan deep learning untuk melakukan deteksi kantuk secara real-time. Metodologi yang diusulkan menggunakan arsitektur jaringan convolutional neural network yang ringan namun akurat versi terbaru yaitu MobileNetV3, bersamaan dengan kerangka kerja Single-Shot Multibox Detector (SSD). Hasil dari penelitian menggunakan MobileNetV3-SSD ini menunjukkan mAP sebesar 0,878 untuk mendeteksi wajah, mata terbuka, dan mata tertutup dan mampu mencapai FPS rata-rata sebesar 15 pada data video secara realtime.

Deteksi kantuk dengan gejala kelelahan seperti menguap, menutup mata atau kenaikan alis pada penelitian (Chmielińska dkk., 2018) menggunakan AlexNet. Hasil terbaik diperoleh untuk detektor menguap. Hasil perhitungan yang dilakukan menunjukkan bahwa sangat mungkin menerapkan pendekatan ini untuk deteksi gejala kelelahan. Namun nilai misclassification rates keseluruhan untuk klasifikasi gejala kurang dari 5,5% merupakan hasil yang cukup memuaskan.

Untuk memberikan analisis yang lebih luas, penulis membandingkan dan mengevaluasi model terhadap arsitektur MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0. Model dievaluasi menggunakan *confusion matrix* untuk mendapatkan *best model*. Pengujian model juga dilakukan dengan mengimplementasi *best model* ke sistem deteksi kelelahan yang dikembangkan oleh Ashish Kushwaha di GitHub-nya. Pada sistem deteksi tersebut, diperlukan model dengan format h5 dan kamera video sebagai *input*. Model harus memiliki 2 *output class* yaitu 0 (mata tertutup) dan 1 (mata terbuka). Gambar 1.1 menyimpulkan cara sistem ini mengidentifikasi dan mendeteksi kelelahan. *Score prediction* adalah nilai prediksi untuk *output* 0 dan 1 saat mata terdeteksi. Deteksi dilakukan pada setiap *frame* gambar video. Setiap mata terbuka atau tertutup akan mengubah *score prediction* tiap *frame.* Jika mata tertutup maka nilai *score prediction* bertambah 1, sebaliknya jika mata terbuka maka nilai *score prediction* berkurang 1 hingga kembali ke 0. Sehingga, seseorang dikatakan berada pada kondisi kelelahan jika mata tertutup lebih 15.

$$Score\ Prediction\ >\ 15\ =\ Indikasi\ Mata\ Lelah$$

Gambar 1.1 Gambaran sistem pendeteksi kelelahan yang digunakan

1.2 Rumusan Masalah

Berdasarkan latar belakang dan penelitian sebelumya, sudah pernah ada penelitian tentang klasifikasi kelelahan menggunakan metode CNN berdasarkan fitur mata, mulut atau wajah. Akan tetapi, model yang dihasilkan memiliki ukuran yang besar. Oleh sebab itu perlu membangun model yang berukuran kecil dan bisa diimplementasi ke sistem pendeteksi kelelahan menggunakan perangkat dengan *resource* terbatas.

1.3 Tujuan Penelitian

1. Mengetahui model terbaik dalam proses klasifikasi kelelahan berdasarkan kondisi mata antara MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0.
2. Mengetahui hasil implementasi model dalam mendeteksi mata tertutup dan terbuka pada sistem pendeteksi kelelahan mata.

1.4 Batasan Masalah

1. Penelitian menggunakan gambar mata dengan kondisi tertutup dan terbuka dari 32 subjek yang diantaranya ada yang menggunakan kacamata gelap dan kacamata baca serta berada diruang kurang cahaya dan terang.
2. Penelitian ini menggunakan data dari MRL Eye Dataset yang dapat di unduh http://mrl.cs.vsb.cz/eyedataset

1.5 Manfaat Penelitian

Penelitian ini memiliki manfaat sebagai berikut:

1. Sebagai acuan bagi penelitian tentang implementasi CNN pada proses klasifikasi
2. Sebagai acuan atau bahan rujukan untuk penelitian selanjutnya tentang klasifikasi kelelahan.
3. Penambah informasi tentang klasifikasi kelelahan berdasarkan kondisi mata.
4. Menjadi sebuah referensi ilmiah yang berguna dalam kesehatan dan psikologi.

1.6 Sistematika Penulisan

1. Abstrak berisi rangkuman isi laporan secara umum.
2. Pernyataan Keaslian Tulisan.
3. Daftar Publikasi berisi jurnal internasional.
4. Halaman Kontribusi berisi dosen yang terlibat dalam penelitian.

5. Halaman Persembahan berisi ucapan terhadap keluarga.
6. Kata Pengantar berisi ucapan terima kasih kepada berbagai pihak yang terlibat dalam penelitian.
7. Daftar Isi berisi daftar judul bab, dan sub bab laporan.
8. Daftar Tabel berisi daftar nama tabel.
9. Daftar Gambar berisi daftar nama Gambar.
10. BAB 1 Pendahuluan
    a. 1.1 Latar Belakang berisi alasan penulis memilih tema tersebut.
    b. 1.2 Rumusan Masalah berisi pertanyaan penelitian.
    c. 1.3 Tujuan Penelitian berisi tujuan penelitian.
    d. 1.4 Batasan Masalah berisi batasan apa saja yang akan dibahas dalam penelitian.
    e. 1.5 Manfaat Penelitian berisi manfaat yang didapatkan sesudah melakukan penelitian.
    f. 1.6 Sistematika Penulisan berisi daftar penulisan dari BAB awal sampai akhir.
11. BAB 2 Tinjauan Pustaka
    a. Penelitian Terdahulu berisi penelitian yang dilakukan sebelumnya.
    b. Konsep Pengetahuan Model berisi model apa saja yang dipakai dalam penelitian.
12. BAB 3 Metodologi
    a. 3.1 Tahapan Penelitian berisi tahapan yang dilakukan dalam penelitian secara urut.
    b. 3.2 Analisis kebutuhan dalam melakukan klasifikasi model.
13. BAB 4 Hasil Dan Pembahasan
    a. 4.1 Dataset yang digunakan
    b. 4.2 Tahapan *Training*
    c. 4.3 *Model Evaluation*
    d. 4.4 Impelementasi dan pengujian model pada sistem deteksi kelelahan
    e. 4.5 *Source Code Training* berisi code python yang dilakukan dalam *Transfer Learning*.
    f. 4.6 *Source Code Matrik Confusion* berisi pengukuran performa.
14. BAB 5 Kesimpulan Dan Saran

a. 5.1 Kesimpulan berisi kesimpulan yang didapatkan sesudah melakukan penelitian.
b. 5.2 Saran berisi saran yang dapat dilakukan pada penelitian selanjutnya.

15. Daftar Pustaka berisi rujukan yang dikutip dalam laporan.
16. Lampiran berisi bukti permintaan ijin penggunaan dataset.

BAB 2
Tinjauan Pustaka

2.1 Penelitian Terdahulu

Penelitian terhadap klasifikasi kelelahan telah banyak dilakukan dengan menggunakan berbagai macam arsitektur dan model. Beberapa penelitian mengklasifikasikan kelelahan berdasarkan mulut yang terbuka menguap dan tertutup, kondisi mata saja ataupun keduanya. Implementasi klasifikasi tersebut juga telah dilakukan ke sebuah sistem pendeteksi kelelahan menggunakan komputer dengan kemampuan *resource* yang cukup besar, sehingga penelitian masih dilakukan agar mendapatkan model yang cukup ringan sehingga lebih mudah di implementasi ke sebuah sistem yang cukup ringan seperti *mobile app.*

Deep Learning adalah bidang pembelajaran mesin yang terinspirasi oleh otak manusia dan upaya untuk memodelkan jaringan saraf menggunakan pemodelan abstrak menggunakan serangkaian fungsi non-transformasi yang ditumpuk secara linier berlapis-lapis (Feriawan dkk., 2020). Perkembangan bidang Deep Learning saat ini telah dipermudah oleh banyaknya library dan Application Program Interface (API). Library yang digunakan adalah Tensorflow yang merupakan antarmuka untuk mengekspresikan algoritma pembelajaran mesin dan untuk mengeksekusi perintah dengan menggunakan informasi yang dimiliki tentang objek tersebut atau target yang dikenali serta dapat membedakan objek satu dengan objek lainnya. Tensorflow memiliki fitur untuk menjalankan pelatihan model menggunakan *Central Processing Unit* (CPU) dan pelatihan model *Graphic Processing Unit* (GPU). Namun dalam penelitian ini akan dijalankan pelatihan model dengan fitur GPU.

CNN merupakan salah satu metode Deep Learning. Arsitektur CNN terdiri 5 bagian utama yaitu *input layer, convolution layer, pooling layer, fully connected layer dan output* layer (Howard dkk., 2020). *Convolution layer* merupakan operasi konvolusi anatara 2 vektor. Pada persamaan (1) merupakan konvolusi dua buah fungsi dimana g(x) disebut sebagai kernel konvolusi (filter) yang akan dioperasikan secara bergeser pada vektor F(x).

Penelitian ini digunakan metode CNN yaitu MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0. MobileNet sesuai dengan namanya, model ini dikembangkan untuk membuat arsitektur CNN dapat digunakan untuk perangkat *mobile* atau yang memiliki *resource* terbatas. EfficientNet lebih cocok untuk perangkat seluler dengan memperkenalkan fungsi aktivasi ReLU6 dan menghapus blok pemerasan dan eksitasi (Tan dkk., 2019).

MobileNet merupakan sebuah model arsitektur CNN yang didesain efisien dengan 2 set *hyper-parameters* untuk membangun model yang sangat kecil dan latensi rendah yang akan dengan mudah diimplementasikan sesuai kebutuhan *mobile* dan *embedded applications*. MobileNet dibuat berdasarkan *depthwise separable convolutions* untuk mengurangi komputasi di layer awal (Santoso dkk., 2018).

EfficientNet merupakan singkatan dari Efficient Network yang dikembangkan oleh Architecture Deep Learning Model MNasNet, namun telah ditingkatkan dalam hal optimalisasi dan akurasi model ukuran. Jika dibandingkan dengan ConvNet lain dengan akurasi yang sama, model EfficientNet memerlukan lebih sedikit parameter dan FLOPS (Floating Points Performance Per Second) (Ab Wahab dkk., 2021).

Penelitian (Khursheed dkk., 2022) telah membuktikan ResNet50V2 memiliki performa yang lebih baik sebesar 97.5% dibandingkan model pre-trained lain seperti InceptionV3, VGG16 dan MobilenetV2. Transfer learning dalam metode ini mengambil lapisan pre-trained model CNN seperti VGG16, ResNet50V2, InceptionV3 dan MobileNetV2, kemudian menggabungkannya dengan tiga lapisan fully connected layers yang dimodelkan sendiri.

Pada penelitian ini, MobileNetV1, MobileNetV3 Large dan EfficientNet Lite digunakan untuk menghasilkan *best model* dengan ukuran yang lebih kecil, sehingga dapat di implementasi ke sebuah sistem pendeteksi kelelahan pada *device* yang memiliki *resource* terbatas seperti *dashboard cam* mobil atau di laptop spesifikasi rendah yang memiliki *webcam*.

Pada Tabel 2.1 menjabarkan berbagai penelitian sebelumnya mengenai klasifikasi dan Deep Learning. Bidang atau Tema penelitian merupakan berbagai kumpulan acuan untuk menyelesaikan penelitian ini. Sedangkan keterangan atau alat analisis merupakan inti sari dari analisis pengembangan riset yang dilakukan pada penelitian yang terkait dari setiap penelitian yang menjadi *state of the art*.

Tabel 2.1 *State of the Art* penelitian

Peneliti	**Bidang/Tema**	**Keterangan / Alat Analisis**
(Suresh dkk., 2021)	Deep Learning untuk deteksi kantuk	Mendeteksi pengemudi mengantuk dengan mengklasifikasikan kondisi mata pengemudi, yaitu terbuka atau tertutup menggunakan InceptionV3.

(Adianto, F. K., 2021)	Deep Learning untuk deteksi kantuk	Mendeteksi kantuk secara *real-time* menggunakan MobileNetV3.
(Chmielińska, J., & Jakubowski, J. M., 2018)	Deep Learning untuk deteksi kantuk	Mendeteksi kantuk dengan gejala kelelahan seperti menguap, menutup mata atau kenaikan alis menggunakan AlexNet.
(Khursheed dkk., 2022)	Training model untuk proses klasifikasi kelelahan pada mata.	Pengembangan model klasifikasi kelelahan menggunakan VGG16, ResNet50V2, InceptionV3 dan MobileNetV2.
Penelitian yang akan dikembangkan	Klasifikasi kondisi mata untuk deteksi kelelahan.	Fokus penelitian ini adalah mendapat *best model* yang memiliki ukuran yang kecil menggunakan MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0.

2.2 Konsep Pengetahuan Model yang Digunakan

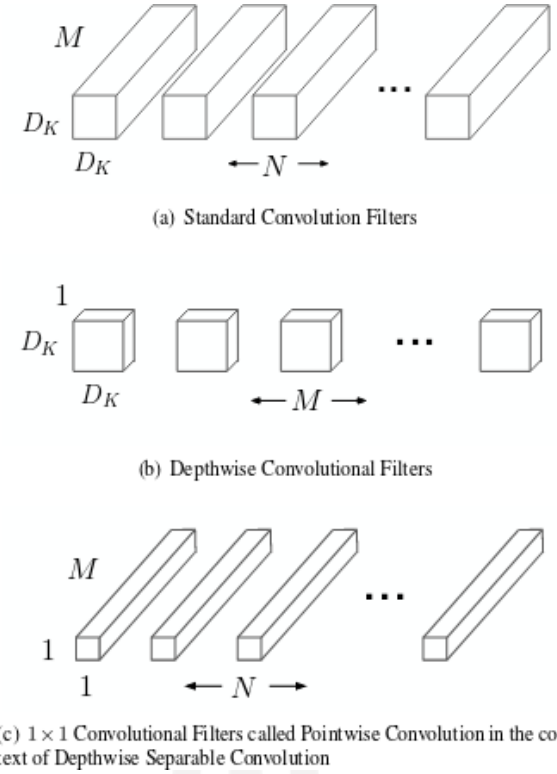
Konsep pengetahuan model menggunakan TensorFlow keras. Konsep yang dimaksudkan adalah pembuatan model. Aplikasi dari Keras yang digunakan dalam pembuatan model adalah MobileNet dan EfficientNet Lite.

2.2.1 MobileNet

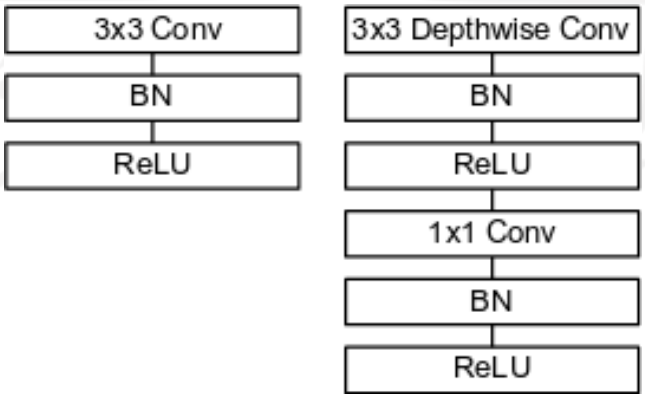
MobileNet sesuai dengan namanya, model ini dikembangkan untuk membuat arsitektur CNN dapat digunakan untuk perangkat *mobile* atau yang memiliki *resource* terbatas. MobileNet merupakan sebuah model arsitektur CNN yang didesain efisien dengan 2 set *hyper-parameters* untuk membangun model yang sangat kecil dan latensi rendah yang akan dengan mudah diimplementasikan sesuai kebutuhan *mobile* dan *embedded applications*. MobileNet dibuat berdasarkan *depthwise separable convolutions* untuk mengurangi komputasi di layer awal (Howard dkk., 2017). Perbedaan mendasar antara arsitektur MobileNet dan arsitektur CNN pada umumnya adalah penggunaan lapisan atau layer konvolusi dengan ketebalan filter yang sesuai dengan ketebalan dari *input image*.

a. MobileNetV1

MobileNet membagi konvolusi menjadi *depthwise convolution* dan *pointwise convolution* seperti pada gambar 2.1 menjelaskan konvolusi standard (a) dibagi menjadi dua lapisan: depthwise convolution (b) dan pointwise convolution (c) untuk membuat filter terpisah secara mendalam (depthwise) dan gambar 2.2. Arsitektur MobileNet sendiri dipaparkan pada gambar 2.3.



Gambar 2.1 Pembagian konvolusi MobileNet



Gambar 2.2 Konvolusi MobileNet

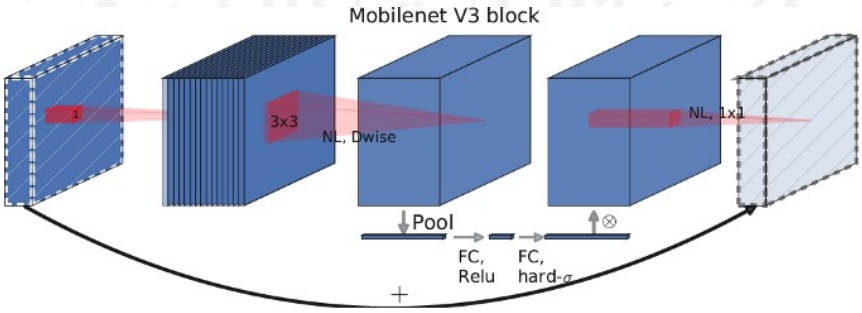
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Gambar 2.3 Arsitektur MobileNetV1

b. MobileNetV3 Large

MobileNetV3 dirancang untuk CPU ponsel melalui kombinasi *hardware-aware network architecture search* (NAS), yaitu MnasNet yang dilengkapi dengan algoritme NetAdapt. Dibandingkan dengan versi MobileNet lain, MobileNetV3 telah memasukkan modul Squeeze and Excitation (SE) yang berasal dari SENet. Pada MobileNetV3 terdapat kombinasi *hidden layer* yang efektif. Salah satunya bagian pada layer tersebut menggunakan modifikasi *swish non linearities*, untuk lebih jelasnya dapat dilihat pada Gambar 2.4.



Gambar 2.4 Susunan *hidden layers* MobileNetV3

Fungsi non linier atau *swish* digunakan sebagai pengganti *drop-in* untuk ReLU yang secara signifikan meningkatkan akurasi neural network. Arsitektur MobileNet sendiri dipaparkan pada gambar 2.5.

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Gambar 2.5 Arsitektur MobileNetV3 Large

MobileNetV3-Large dilatih untuk ditargetkan pada penggunaan sumber daya tinggi dan rendah, seperti yang ditampilkan di atas. Model dibuat melalui penerapan *platform-aware* NAS (MnasNet) dan NetAdapt untuk pencarian jaringan dan menggabungkan peningkatan jaringan.

2.2.2 EfficientNet Lite

EfficientNet-Lite menjadikan EfficientNet cocok untuk perangkat *mobile* dengan memperkenalkan fungsi aktivasi ReLU6. Model EfficientNet-lite memiliki lima varian yaitu lite0, lite1, lite2, lite3 dan lite4. EfficientNet-lite adalah sekumpulan model klasifikasi gambar yang di optimalisasi untuk seluler/IoT. Meskipun EfficientNet-EdgeTPU dikhususkan untuk Coral EdgeTPU, tetapi model EfficientNet-lite ini berjalan dengan baik di semua CPU/GPU/EdgeTPU seluler.

Gambar 2.6 dan Gambar 2.7 menunjukkan bagaimana model EfficientNet-Lite terkuantisasi bekerja dibandingkan dengan versi terkuantisasi serupa dari beberapa model klasifikasi gambar populer. EfficientNet Lite0 memiliki model size terkecil dari versi EfficientNetLite lainnya sehingga ini menjadikan alasan memilih model ini digunakan dalam pemodelan sesuai tujuan penelitian, meskipun secara akurasi akan turun.

Gambar 2.6 Hasil ukuran model (mb) dan akurasi EfficientNet-Lite

Gambar 2.7 Hasil ukuran model (mb) dan akurasi EfficientNet-Lite

2.2.3 Kelelahan Mata

Kelelahan mata adalah ketegangan pada mata yang disebabkan oleh penggunaan indera penglihatan dalam bekerja yang memerlukan kemampuan untuk melihat dalam jangka waktu yang lama dan biasanya disertai dengan kondisi pandangan yang tidak nyaman (Abtahi dkk., 2011). Untuk menentukan kondisi kelelahan seseorang saat belajar adalah dengan mengukur durasi lamanya kondisi mata tertutup. Pada umumnya setiap orang ketika mulai lelah dan

mengantuk jarak antara kedua kelopak mata semakin menyempit dan frekuensi kedipan semakin menurun hingga mata cenderung tertutup lebih lama dari pada mata terbuka.

Kedipan mata adalah ketika seseorang menutup dan membuka kembali. Setiap individu memiliki pola perbedaan yang berbeda. Polanya berbeda dalam kecepatan menutup dan membuka, tingkat kedipan mata dan dalam durasi kedipan. Mata berkedip terjadi di sekitar 100-400 ms (Caffier dkk., 2005). Berdasarkan penelitian tersebut, kelelahan seseorang dapat dilihat dari kondisi mata tertutup dan terbuka.

Dengan mengumpulkan dan menggunakan gambar dari mata dengan kondisi tertutup dan terbuka, dapat dilakukan pengembangan model klasifikasi kondisi mata menggunakan metode CNN. Metode CNN merupakan suatu kelas pada *neural network* yang berspesialisasi dalam memproses data yang memiliki topologi seperti grid, misalnya gambar. CNN mampu menangkap dependensi spasial dan temporal suatu gambar melalui operasi konvolusi dengan filter/kernel (Valueva et al., 2020). Oleh karena itu, CNN dapat dilatih untuk memahami detail sebuah gambar mata terbuka dan tertutup dengan lebih baik.

Dalam penelitian ini, peneliti hanya berfokus mengembangkan model CNN untuk klasifikasi mata tertutup dan terbuka. Namun, hasil klasifikasi model akan di gunakan pada studi kasus kelelahan mata. Implementasi model digunakan untuk mendeteksi kelelahan dalam ruang dan waktu dalam rangkaian video melalui *webcam* laptop. Fitur gerak yang diekstraksi oleh model CNN spasial-temporal peneliti gunakan untuk secara efektif dan kuat menangkap anomali gerak mata untuk berbagai skenario.

BAB 3
Metodologi

3.1 Tahapan Penelitian

Pengumpulan data, pelatihan data dan evaluasi hasil klasifikasi merupakan tiga langkah utama yang dilakukan pada penelitian ini. Namun, secara umum tahapan tersebut akan dijelaskan seperti Gambar 3.1.

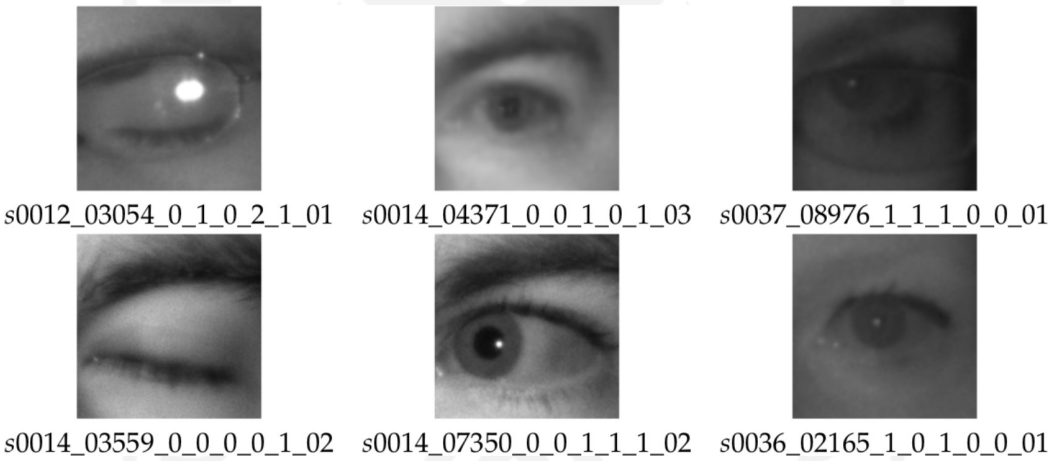
Gambar 3.1 Tahapan penelitian

3.1.1 Data Gathering

Tahapan awal yang dilakukan adalah menyiapkan dataset yang berupa himpunan gambar kondisi mata. Dataset digunakan sebagai input untuk melakukan pelatihan model. Pada penelitian ini digunakan dataset MRL Eye Dataset, didalamnya terdapat gambar *infrared* di resolusi rendah dan tinggi yang diambil dari 37 subjek yang berbeda dan telah dibagi dalam beberapa kategori. Dataset ini memiliki delapan anotasi yakni *subject ID*, *image number, gender, glasses, eye state, reflections, lighting conditions,* dan *sensor type*. Gambar 3.2 menunjukkan contoh dataset yang digunakan telah memiliki label berdasarkan delapan anotasi. Setiap anotasi pada label dataset dipisahkan dengan karakter garis bawah. Misalkan pada gambar mata pertama memiliki label s0012_03054_0_0_2_1_01 dengan penjelasan sebagai berikut:

- s0012 untuk anotasi *subject ID* artinya orang tersebut diurutan ke 12. *Subject* terdiri dari 33 pria dan 4 wanita.
- 03054 untuk anotasi *image number* artinya gambar ini diurutan 3054. Total jumlah *image* adalah 84.898.
- 0 untuk anotasi *gender*. Anotasi ini memiliki label 0 untuk pria dan 1 untuk wanita.

- 0 untuk anotasi *eye state.* Anotasi ini memiliki label 0 untuk mata tertutup dan 1 untuk mata terbuka.
- 2 untuk anotasi *reflections*. Anotasi ini memiliki tiga label sesuai tingkatan *reflections* yaitu 2 untuk *big reflections,* 1 untuk *small reflections* dan 0 jika tanpa *reflections.*
- 1 untuk anotasi *lightning conditions*. Anotasi ini memiliki 2 label sesuai dengan kondisi cahaya pada gambar, label 1 untuk gambar yang memiliki *good lightning* dan label 0 untuk gambar yang memiliki *bad lightning* misal gambarnya gelap.
- 01 untuk anotasi *sensor ID.* Dataset di ambil dari tiga jenis sensor dengan perbedaan resolusi. Label 01 untuk gambar yang diambil dengan sensor RealSense RS 300 dan resolusinya 640x480, label 02 untuk gambar yang diambil dengan sensor IDS Imaging Sensor dan resolusinya 1280x1024, label 03 untuk gambar yang diambil dengan sensor Aptina dan resolusinya 752x480.



Gambar 3.2 Contoh dataset MRL Eye Dataset

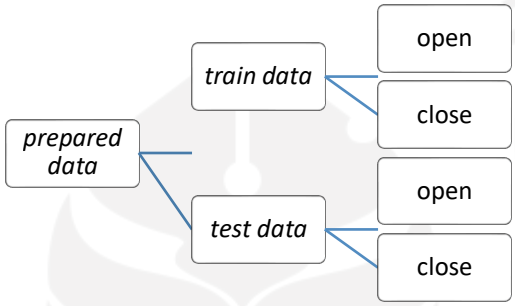
Pada penelitian ini, dari 8 anotasi yang ada pada dataset hanya anotasi *eye state* yang digunakan. Peneliti hanya menggunakan anotasi *eye state* dikarenakan sistem kelelahan yang akan digunakan untuk menguji model menggunakan 2 *class* saja. *Eye state* adalah anotasi untuk kondisi mata subjek yang tertutup dan terbuka. Sehingga hanya anotasi ini yang dibutuhkan untuk menghasilkan model hasil klasifikasi mata terbuka dan mata tertutup.

Pada Tabel 3.1 menunjukkan keterangan anotasi mata pada dataset yang memiliki 2 label, yaitu label 0 untuk anotasi mata tertutup dan label 1 untuk anotasi mata terbuka yang akan digunakan dalam klasifikasi model.

Tabel 3.1 Keterangan Anotasi

Kondisi	Label	Anotasi
Mata	0	Tertutup
	1	Terbuka

Berdasarkan label yang sudah ada, *pre-processing data* dilakukan dengan membagi dataset yang memiliki label 0 ke folder *closed* dan dataset yang memiliki label 1 ke folder *open.* Kemudian *prepared data* disiapkan untuk proses pelatihan, sehingga data akan dibagi menjadi *data train* dan *data test*. Gambar 3.3 adalah struktur folder *prepared data*.



Gambar 3.3 Struktur folder *prepared data*

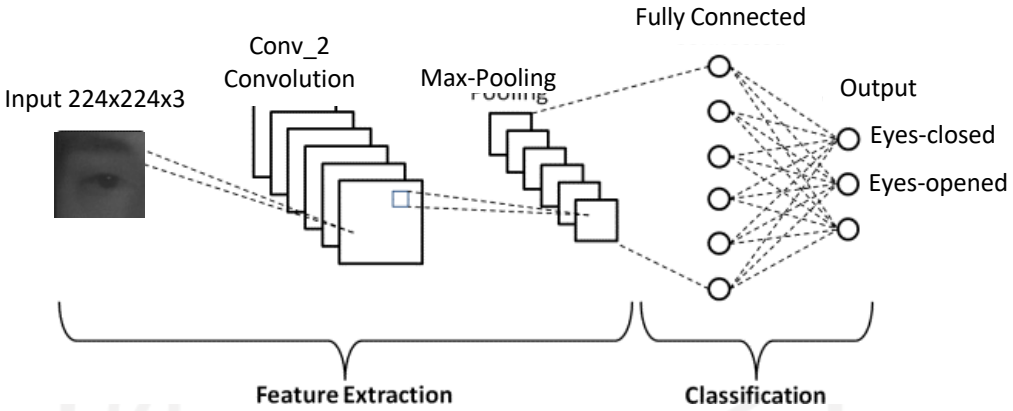
Prepared data adalah dataset yang telah siap digunakan pada proses *training* yang berisikan data yang sudah memiliki label. Pada *train data,* jumlah data yang memiliki label 1 di folder *open* berjumlah 42952 gambar dan jumlah data yang memiliki label 0 di folder *close* berjumlah 41946. Pada *test data*, jumlah data yang disiapkan sebagai data tes adalah 2554 untuk masing-masing label.

3.1.2 Data Training

Data training adalah tahap pelatihan data untuk melatih dan mengembangkan model. *Data train* akan dilatih dengan menggunakan metode *convolutional neural network,* sedangkan *data test* akan digunakan untuk menguji model yang dihasilkan. Proses *training* ini merupakan tahapan dimana CNN dilatih untuk memperoleh akurasi yang tinggi dari klasifikasi yang dilakukan (Putra, 2016).

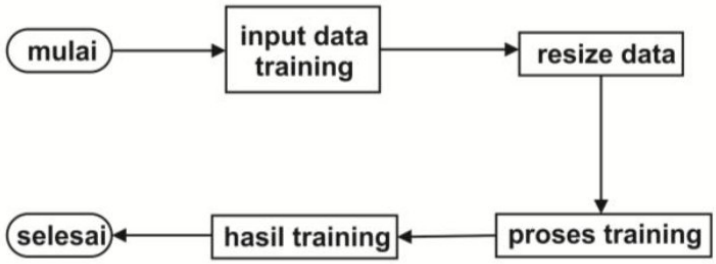
Gambar 3.4 merupakan proses CNN pada klasifikasi mata tertutup dan terbuka. Gambar mata yang menjadi input proses training di *resize* menjadi 224x224 pixel.

Klasifikasi ini dilakukan untuk menghasilkan *output* mata tertutup (*eyes closed)* dan mata terbuka (eyes-opened).



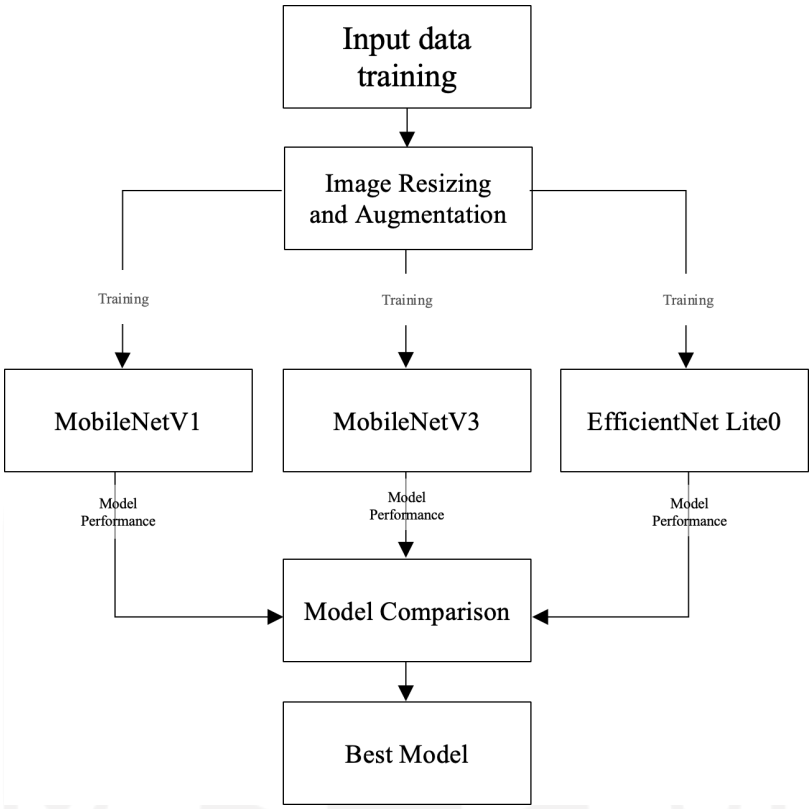
Gambar 3.4 Proses *Convolusional Neural network*

Tujuan dari proses *training* adalah melatih model jaringan syaraf tiruan untuk meminimalkan error hasil prediksi model dengan data asli. Gambar 3.5 menunjukan alur secara umum yang akan dilakukan berulang dengan menggunakan tiga aristektur *mobile* CNN.



Gambar 3.5 Alur proses training

Penelitian ini menggunakan tiga *base model* dari arsitektur *mobile* CNN, yaitu MobileNetV1, MobileNetV3 dan EfficientNet Lite0. Proses *training* dilakukan untuk masing-masing *base model* dengan *input data training* yang sama. Gambar 3.6 menjelaskan tahapan awal proses *training* yang dilakukan hingga mendapatkan *best model* yang diinginkan.



Gambar 3.6 Tahapan klasifikasi MobileNetv1, Mobilenetv3 dan Efficientnet lite0

Langkah awal adalah input data *training.* Pada tahap ini, data yang telah dilabeli dan dibagi menjadi *data train dan data validation* akan digunakan dalam proses pelatihan. Setelah itu, mengatur *hyperparameter* seperti *batch size, learning rate* dan *image size*. Selanjutnya dilakukan *augmentation.* Augmentasi data adalah tindakan manipulasi data gambar sedemikian rupa sehingga komputer mengenali gambar yang diperbarui sebagai gambar baru tetapi orang masih dapat mengidentifikasinya sebagai gambar yang sama (Wang dkk., 2017). Tahapan *training* dilakukan dengan menggunakan *base model* MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0. Pada tahap *model comparison,* ketiga model hasil dari pemrosesan tersebut akan dievaluasi dan diuji, sehingga didapatkan *best model* yang akan digunakan untuk mengklasifikasi kelelahan pada mata.

3.1.3 Model Evaluation

Ada beberapa faktor yang dibutuhkan untuk menentukan model terbaik yaitu dari performa matrix dari hasil training model yang telah dibuat, perbandingan hasil matrik konfusi dari model yang telah dibuat dan *inference time* model yang telah dibuat. Dari evaluasi model

akan diketahui model terbaik dalam klasifikasi mata tertutup dan lelah. Selanjutnya model terbaik tersebut akan diuji pada sistem deteksi kelelahan.

3.2 Analisis Kebutuhan

Analisis kebutuhan adalah metode untuk memperoleh detail dan spesifikasi peralatan berfokus pada perangkat lunak yang akan dibuat oleh penulis. Tujuan dari analisis kebutuhan adalah untuk menentukan kebutuhan sistem dalam klasifikasi pada kelelahan berdasarkan kondisi mata. Spesifikasi ini mencakup kebutuhan masukan, kebutuhan perangkat lunak, dan kebutuhan keluaran.

3.2.1 Analisis Kebutuhan Masukan

Kebutuhan masukan pada proses klasifikasi kelelahan berdasarkan kondisi mata adalah gambar hasil ekstraksi *video frame* yang berekstensi png yang telah diunduh dari sumber http://mrl.cs.vsb.cz/eyedataset

3.2.2 Analisis Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak untuk proses klasifikasi kelelahan adalah Jupyter Notebook yang digunakan untuk menjalankan sistem pendeteksi kelelahan dengan menggunkan model ke sistem.

3.2.3 Analisis Kebutuhan Keluaran

Kebutuhan keluaran untuk proses klasifikasi kelelahan adalah teks dan persentase yang menunjukan berapa persen yang indentifikasi mata tertutup dan terbuka.

BAB 4
Hasil dan Pembahasan

Pada hasil dan pembahasan penulis memaparkan proses lebih rinci untuk mendapatkan hasil pembuatan model dari proses *training*, hasil matrik konfusi, cara untuk mengukur waktu yang dibutuhkan untuk memperoleh hasil tiap model. Model terbaik diantara MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0 akan di implementasi ke sistem pendeteksi kelelahan yang sudah ada.

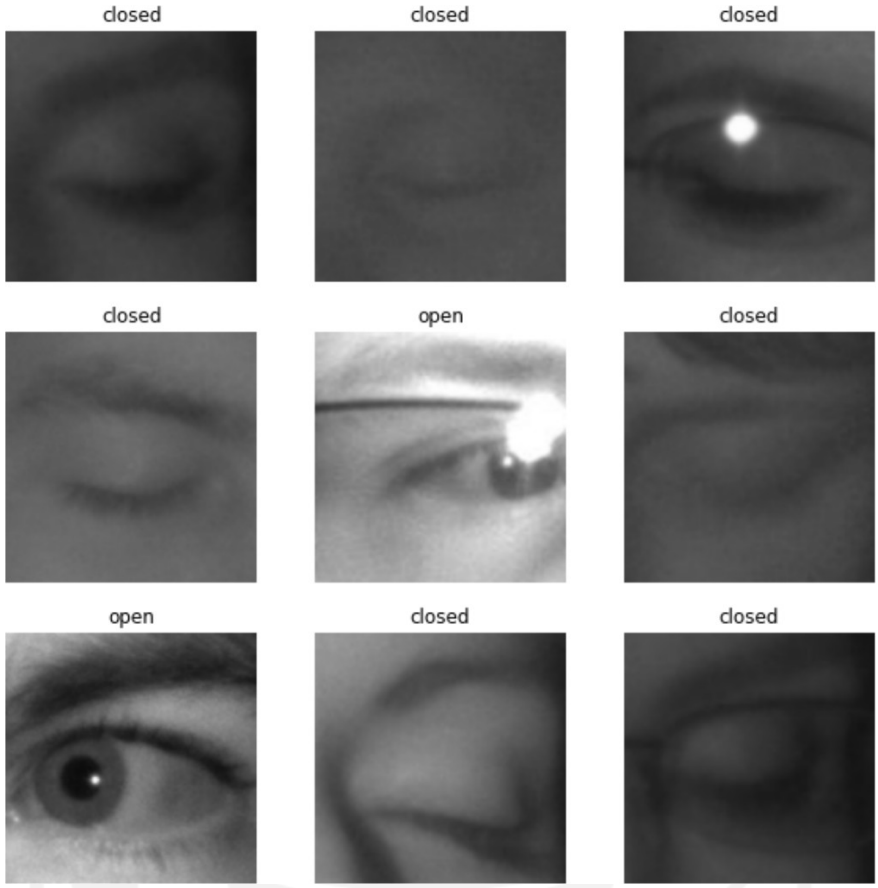
4.1 Dataset yang digunakan

Dataset yang digunakan adalah MRL Eye dataset. Dataset ini berjumlah kurang lebih 84.900 dan sudah memiliki label berdasarkan 8 anotasi. Dikarenakan hanya anotasi *eye state* yang digunakan, maka perlu dilakukan *pre-processing* data seperti Gambar 4.1. *Pre-processing* dilakukan dengan membagi dataset ke folder *closed* untuk dataset yang memiliki anotasi *eye state* dengan label 0 dan folder *opened* untuk dataset yang memiliki anotasi *eye state* dengan label 1.

```
raw_data = '/home/delixus/Desktop/drowsiness_detection/mrlEyes_2018_01'
for dirpath, dirname, filename in os.walk(raw_data):
    for file in tqdm([f for f in filename if f.endswith('.png')]):
        if file.split('_')[4] == '0':
            path='/home/delixus/Desktop/drowsiness_detection/data/train/closed'
            if not os.path.exists(path):
                os.makedirs(path)
            shutil.copy(src=dirpath + '/' + file, dst= path)
        elif file.split('_')[4] == '1':
            path='/home/delixus/Desktop/drowsiness_detection/data/train/open'
            if not os.path.exists(path):
                os.makedirs(path)
            shutil.copy(src=dirpath + '/' + file, dst= path)
```

Gambar 4.1 *Split* dataset berdasarkan anotasi *eye state*

Selanjutnya, dataset tersebut digunakan untuk membentuk *data train* dan *data test.* Hanya digunakan 20 persen data untuk menghasilkan 8590 *data train* dan 2554 *data test*. Gambar 4.2 menunjukkan plot dataset yang telah diberi *class name close* untuk dataset dengan label 0 dan *class name open* untuk dataset dengan label 1.



Gambar 4.2 Label *open* dan *closed* pada gambar

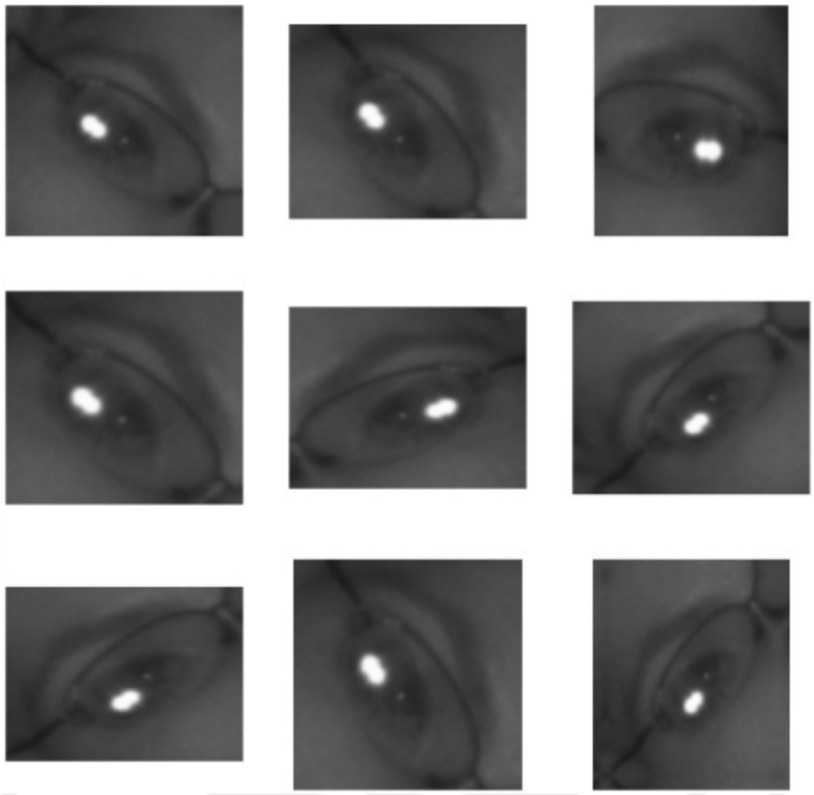
4.2 Tahapan *training*

Tahapan *training* merupakan proses pemodelan yang dilakukan dengan teknik pembelajaran dan pelatihan gambar mata untuk proses klasifikasi agar data yang telah dimodelkan bisa menentukan apakah mata seseorang tertutup atau terbuka. Prosedur ini dilakukan pada MSI GF63 Thin 10SC dengan spesifikasi:

- Prosesor: Intel Core i5-10500H (2.50GHz)
- RAM : 24 GB DDR3L
- NVDIA GEFORCE GTX

Dengan menggunakan 8590 *data train* dan 2554 *data test,* langkah awal dilakukan dengan mengatur *hyperparameter* sebelum memulai proses training. *Batch size* diatur menjadi 32 agar sistem tidak terlalu berat melakukan *training*. Dataset yang digunakan memiliki *image* berukuran 640x480 pixel, ini menjadi ukuran yang sangat besar dan akan menyebabkan sistem terlalu berat untuk melakukan proses pelatihan, maka dilakukan *resize image* menjadi 224x224. Selanjutnya, pada proses augmentasi gambar akan mengalami rotasi sekitar 20 derajat. Selain rotasi, augmentasi lain yang perlu dilakukan adalah

mengubah tinggi, lebar dan memperbesar gambar sebesar 20 persen serta membalik gambar secara horizontal. Hasil augmentasi yang dilakukan dapat dilihat pada Gambar 4.3.



Gambar 4.3 Hasil augmentasi gambar mata

Augmentasi data merupakan tindakan mengubah atau memanipulasi gambar sedemikian rupa sehingga komputer mengenali gambar yang diperbarui sebagai gambar baru tetapi masih dapat mengidentifikasi itu adalah gambar yang sama (Wang & Perez, 2017). Gambar yang telah diaugmentasi akan digunakan sebagai masukan pada metode CNN untuk melakukan klasifikasi mata tertutup dan mata terbuka, hal ini dilakukan agar data lebih variatif. Selanjutnya, gambar yang telah diekstraksi akan siap untuk memasuki proses *training* masing-masing menggunakan *base model* MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0.

Pada proses *training* masing-masing *base model* dilakukan dengan nilai *learning rate* diatur sebesar 0,0001 dan epoch 50, yang dibagi menjadi 25 epoch tanpa *fine tuning* dan 25 epoch dengan *fine tuning*. Tujuan *fine tuning* adalah melatih *pre-trained* model pada *top-layer* untuk mengenali fitur dari dataset yang dimiliki. Semakin besar jumlah *epoch* dan semakin kecil *learning rate* maka semakin baik tingkat akurasi pelatihan yang didapatkan.

Hasil proses *training* di gambarkan dengan sebuah plot *accuracy* dan *cross entrophy*. Plot *accuracy* digunakan untuk menampilkan grafik *training validation* dan *validation*

accuracy. Sedangkan plot *cross entrophy* merupakan *loss function* yang digunakan untuk menampilkan grafik *training loss* dan *validation loss*. Adapun hasilnya sebagai berikut:

a. *Hasil* training MobileNetV1

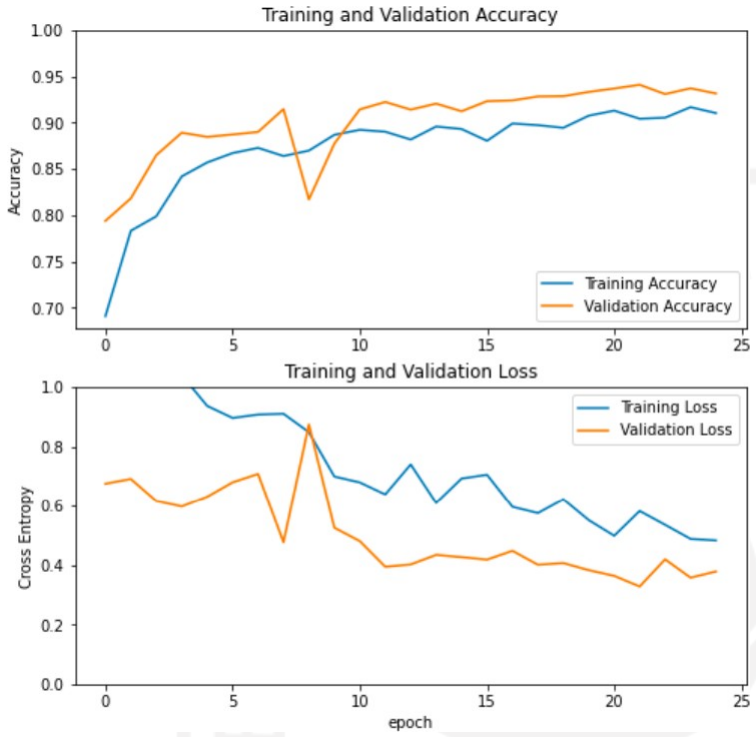
Berdasarkan hasil *training* yang dilakukan menggunakan MobileNetV1, Gambar 4.4 pada plot atas yaitu plot *accuracy,* garis biru menunjukkan nilai *training accuracy* sebesar 0.95 dan garis jingga menunjukkan nilai *validation accuracy* sebesar 0.96. Sedangkan plot bawah yaitu plot *cross entrophy,* garis biru menunjukkan nilai *training loss* sebesar 0.29 dan garis jingga menunjukkan nilai *validation loss* sebesar 0.20. Akurasi dari prediksi model cukup tinggi diatas 90 persen dan prediksi model ini melakukan sedikit kesalahan dengan nilai *loss* dibawah 30 persen, sehingga dapat dikatakan model yang dihasilkan MobileNetV1 sudah cukup bagus.

Gambar 4.4 Hasil plot *accuracy* dan *cross entrophy* MobileNetV1

b. Hasil *training* MobileNetV3 Large

Hasil *training* yang dilakukan menggunakan MobileNetV3 Large seperti Gambar 4.5 memiliki akurasi yang lebih rendah dibandingkan MobileNetV1 yang dimana garis biru menunjukkan nilai *training accuracy* sebesar 0.91 dan garis jingga menunjukkan nilai *validation accuracy* sebesar 0.93. Sedangkan pada plot bawah yaitu plot *cross entophy,* garis biru menunjukkan nilai *training loss* sebesar 0.48 dan garis jingga menunjukkan *validation*

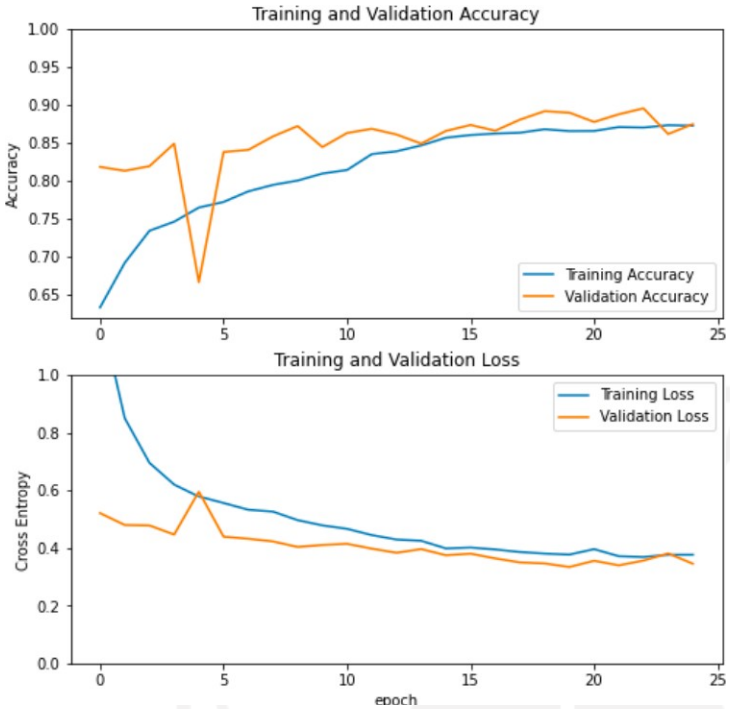
loss sebesar 0.37. Meskipun secara akurasi prediksi model cukup bagus diatas 90 persen dan nilai *loss* cukup kecil, namun prediksi model ini lebih banyak kesalahan diatas 35 persen jika dibandingkan dengan hasil training MobileNetV1.



Gambar 4.5 Hasil plot *accuracy* dan *cross entrophy* MobileNetV3 Large.

c. Hasil *training* EfficientNet Lite0

Berdasarkan hasil *training* yang dilakukan menggunakan EfficientNet Lite0, nilai akurasi prediksi model ini tidak sebaik MobileNetV1 dan MobileNetV3 Large yaitu dibawah 90 persen seperti yang terlihat pada Gambar 4.6, dimana garis biru yang menunjukkan nilai *training accuracy* sebesar 0.87 dan garis jingga menunjukkan nilai *validation accuracy* sebesar 0.87. Akan tetapi prediksi model ini lebih sedikit melakukan kesalahan dibandingkan MobileNetV3 Large yaitu pada plot dibawah garis biru menunjukkan nilai *training loss* sebesar 0.37 dan garis jingga menunjukkan *validation loss* sebesar 0.34. Sehingga model ini tidak sebaik model yang dihasilkan MobileNetV1 dimana secara akurasi lebih tinggi dan kesalahan prediksi model jauh lebih kecil.



Gambar 4.6 Hasil plot *accuracy* dan *cross entrophy* EfficientNet Lite0

Berdasarkan hasil *training* yang dilakukan dengan *learning rate* 0.0001 dan jumlah *epoch* 50, Tabel 4.1 menunjukkan perbandingan *accuracy* dan *loss* dari MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0. Semakin kecil nilai *loss* semakin baik modelnya dan semakin besar nilai *accuracy* semakin baik model dalam prediksi.

Tabel 4.1 Perbandingan *accuracy* dan *loss* model

Architecture	**Batch**	**Validation (%)**	
		Accuracy	**Loss**
MobileNetV1	size 32	96	20
MobileNetV3 Large	size 32	93	37
EfficientNet Lite0	size 32	87	34

Berdasarkan perbandingan *accuracy* dan *loss,* MobileNetV1 menunjukkan *validation accuracy* yang lebih besar dan nilai *loss* yang lebih kecil dari MobileNetV3 Large dan EfficientNet Lite0. Sehingga model yang dihasilkan dengan menggunakan MobileNetV1 menjadi model yang lebih baik dalam hal prediksi dari arsitektur lainnya. Akan tetapi, model harus di evaluasi untuk menentukan *best model* dari ketiga arsitektur tersebut.

4.3 Model Evaluation

Model evaluation adalah tahap dimana model akan dievaluasi. Model dievaluasi menggunakan *confusion matrix* dan dibandingkan sehingga didapatkan *best model*. *Best model* akan digunakan untuk diuji pada sistem pendeteksi kelelahan.

Metrik yang digunakan pada penelitian ini adalah *precision, recall* dan *F1-Score*. *Precision* merupakan nilai presisi atau akurasi prediksi suatu model. *Recall* menunjukkan seberapa banyak *real class* yang ditangkap model. *F1-Score* menjadi metrik yang masuk akal untuk diterapkan jika *precision* dan *recall* perlu diseimbangkan.

Model MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0 dilatih menggunakan 67291 gambar untuk menghasilkan model. Kemudian masing-masing model divalidasi menggunakan 32 gambar. Tabel 4.2 merupakan hasil performa model yang diukur dengan menggunakan Precision, Recall dan F1-score.

- nilai *precision* akan menunjukkan seberapa besar presisi atau akurasi model dalam prediksi
- nilai *recall* menunjukkan jumlah *class* yang ditangkap oleh model
- nilai F1 akan menjadi metrik yang digunakan ketika *precision* dan *recall* perlu diseimbangkan.

Tabel 4.2 Perbandingan Precision, Recall, F1-Score

Architecture	**Batch**	**Class**	**Precision**	**Recall**	**F1-Score**
MobileNetV1	size 32	0	1.00	0.93	0.97
	size 32	1	0.94	1.00	0.97
MobileNetV3 Large	size 32	0	1.00	0.94	0.97
	size 32	1	0.93	1.00	0.97
EfficientNet Lite0	size 32	0	1.00	1.00	1.00
	size 32	1	1.00	1.00	1.00

EfficientNetLite0 memiliki performa yang sangat baik dengan nilai presisi 100 persen pada *class* 0 mata tertutup dan *class* 1 mata terbuka. Akan tetapi, hal ini belum dapat menjadikan model EfficientNet Lite0 menjadi *best model.*

Selanjutnya, dilakukan pengujian konfusi matrik untuk melihat seberapa banyak data yang terdeteksi dengan *true* dan *false* dengan menguji 32 gambar menggunakan

MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0. Tabel 4.3 yang menunjukkan rincian jumlah data yang terdeteksi dengan enam penilaian sebagai berikut:

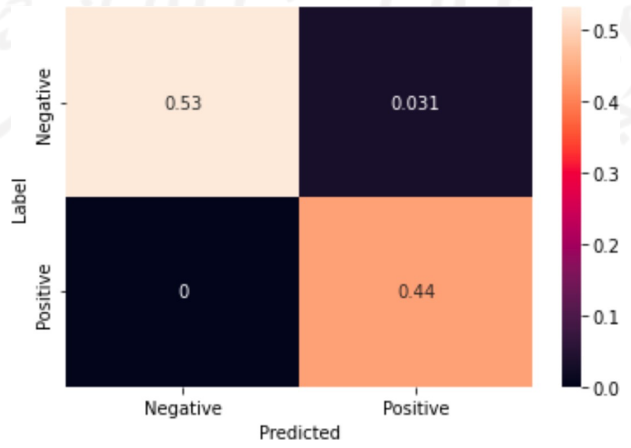
a. *True Positive* (TP) artinya jumlah data yang telah diprediksi tertutup sesuai.
b. *True Negative* (TN) artinya jumlah data yang telah diprediksi terbuka itu sesuai.
c. *False Positive* (FP) artinya jumlah data yang telah diprediksi tertutup tapi tidak sesuai.
d. *False Negative* (FN) artinya jumlah data yang telah diprediksi terbuka tapi tidak sesuai.
e. *Correct Prediction* (CP) artinya jumlah data prediksi yang sesuai.
f. *Incorrect Prediction* (I) artinya jumlah data prediksi yang tidak sesuai.

Tabel 4.3 Hasil Perbandingan TP, FP, TN, FN, CP dan I

Architecture	**Batch**	**TP**	**FP**	**TN**	**FN**	**CP**	**I**
MobileNetV1	size 32	12	2	18	0	30	2
MobileNetV3 Large	size 32	13	0	19	0	32	0
EfficientNet Lite0	size 32	17	1	14	0	31	1

Penentuan *best model* dapat dilakukan dengan melihat jumlah *correct prediction* terbesar dan jumlah *false negative* terkecil. *Correct prediction* (CP) menunjukkan seberapa banyak jumlah prediksi yang sesuai dan *false negative* (FN) menunjukkan seberapa banyak jumlah prediksi yang tidak sesuai. Berdasarkan hal tersebut, meskipun hasil performa metrik EfficientNet Lite0 didapatkan presisi sebesar 1.00, recall 1.00, dan f1-score 1.00. Akan tetapi, MobileNetV3 Large lebih unggul dalam nilai prediksi dengan nilai *false negative* (FN) 0 dan *correct prediction* (CP) 32.

Gambar 4.7 menunjukkan kemampuan model dalam klasifikasi gambar yang divisualisasikan menggunakan heatmap.



Gambar 4.7. *Heatmap* hasil prediksi

Penentuan *best model* juga ditentukan dengan melihat *inference time* sebuah model dalam melakukan prediksi. Semakin kecil *inference time* semakin cepat model melakukan prediksi. Pada Tabel 4.4 menunjukkan hasil perhitungan waktu untuk masing-masing model yakni MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0 dengan batch 32 dengan data validasi 100 gambar.

Tabel 4.4 Perbandingan *inference time*

Architecture	**Batch**	**Test Accuracy**	**Inference Time (s)**
MobileNetV1	size 32	0.98	0.05
MobileNetV3	size 32	0.98	0.05
EfficientNet Lite0	size 32	0.98	0.10

Berdasarkan pengujian *inference time*, MobileNetV1 dan MobileNetV3 Large sama-sama memiliki *inference time* 0.05 detik dan *test accuracy* 0.98. Penelitian ini memiliki tujuan agar menghasilkan model dengan ukuran yang terkecil sehingga lebih ringan untuk di implementasi ke sebuah sistem dengan *device resource* terbatas. Sehingga selain performa model, peneliti juga perlu melihat ukuran model yang dihasilkan. MobileNetV1 memiliki ukuran model paling kecil yaitu 12 MB dibandingkan MobileNetV3 Large sebesar 32 MB dan EfficientNet Lite0 sebesar 24 MB. Oleh karena itu, MobileNetV1 dipilih sebagai *best model* sesuai dengan tujuan yaitu memiliki ukuran terkecil dan meskipun ukurannya kecil tetapi performa model sangat baik karena tidak mengurangi akurasi dengan nilai *test accuracy* yang tinggi yaitu 0.98.

4.4 Impelementasi dan pengujian model pada sistem deteksi kelelahan

Setelah mendapatkan *best model* dengan ukuran terkecil dan melakukan evaluasi *confusion matrix,* penulis juga menggunakan *best model* MobileNetV1 untuk di implementasi dan di uji pada sistem pendeteksi kelelahan yang dikembangkan oleh Ashish Kushwaha di GitHub-nya. Sistem yang dikembangkan menggunakan model h5 yang memiliki model 2 *class* dan dibutuhkan kamera depan untuk mendeteksi mata pada wajah subjek, dalam hal ini kamera *webcam* laptop digunakan.

Model hasil dari penelitian ini memiliki 2 *output class* yaitu 0 untuk mata tertutup dan 1 untuk mata terbuka. Sistem pendeteksi kelelahan ini mengakumulasi nilai *score* dari setiap perubahan nilai dari *output class*.

Gambar 4.8 menunjukkan jika mata terdeteksi tertutup dan memiliki nilai model prediksi lebih dari 0.3, maka *score* ditambah 1 dan jika mata terbuka memiliki nilai model prediksi lebih dari 0.9, maka *score* dikurangi 1 hingga 0. Perubahan nilai *score prediction* dihitung setiap *frame.* Seseorang dikatakan berada pada kondisi kelelahan jika mata tertutup lebih 15 *frame* atau selama 6 detik, sehingga *alarm beep* akan bunyi.

```
# if eyes are closed
if prediction[0][0]>0.30:
    cv2.putText(frame,'closed',(10,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255),
                thickness=1,lineType=cv2.LINE_AA)
    cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255),
                thickness=1,lineType=cv2.LINE_AA)
    Score=Score+1

# if eyes are open
elif prediction[0][1]>0.90:
    cv2.putText(frame,'open',(10,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255),
                thickness=1,lineType=cv2.LINE_AA)
    cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255),
                thickness=1,lineType=cv2.LINE_AA)
    Score = Score-1
    if (Score<0):
        Score=0

if(Score>15):
    try:
        sound.play()
    except:
        pass
```

Gambar 4.8 Deteksi mata lelah dari perubahan *score prediction*

Berikut ini adalah hasil deteksi kelelahan mata menggunakan *best model*. Gambar 4.9 menunjukkan mata terdeteksi terbuka dan memiliki *score* 0. *Score* adalah nilai *score prediction* yang dipengaruhi oleh perubahan pergerakan mata yang terdeteksi. Ketika mata terdeteksi terbuka, maka nilai *score* akan berkurang 1 tiap *frame* hingga berhenti di angka 0. Nilai *score* 0 adalah batas nilai terendah ketika mata terdeteksi terbuka.

Gambar 4.9 Mata terdeteksi terbuka

Ketika mata terdeteksi tertutup seperti pada Gambar 4.10, maka *score* akan bertambah 1 tiap *frame*. Nilai *score* lebih dari 15 akan di identifikasi lelah oleh sistem dan *beep alarm* akan berbunyi.

Gambar 4.10 Mata terdeteksi tertutup

Pengujian model pada sistem pendeteksi kelelahan berhasil membaca kondisi mata subjek melalui *webcam* komputer. Subjek terindikasi lelah dan *alarm* berbunyi karena mata tertutup dengan nilai *score prediction* 21 atau selama 8 detik.

4.5 Source Code Training

Proses *training* dilakukan dengan menggunakan *transfer learning,* dimana penulis menggunakan model yang sudah dilatih sebelumnya dengan cara mengatur mengatur *starting point,* modifikasi parameter sesuai dengan dataset yang digunakan. Pemodelan awal dilakukan menggunakan MobileNetV1, Gambar 4.11 memuat library yang dibutuhkan dan melakukan pengecekan GPU. Penggunaan GPU akan mempercepat proses pemodelan yang akan dilakukan masing-masing pada MobileNetV1, MobileNetV3 Large dan EfficientNet Lite0. Adapun penjelasan terkait *import library* yang dilakukan, yakni:

a. *Import* tensorflow untuk mengaktifkan *library* yang dibutuhkan dalam komputasi saat *train* model klasifikasi dan membuat plot.
b. *Import* MobileNet dari *tensorflow keras application* untuk mengunduh model *pre-train* yang digunakan sebagai *base model.*
c. *Import* Dropout dari *tensorflow keras* untuk mencegah *overfitting*

d. *Import* Flatten dari *tensorflow keras* untuk menggunakan fungsi flatten seperti membuat input yang memiliki banyak dimensi menjadi satu dimensi.
e. *Import* Dense dari *tensorflow keras* untuk menjalankan *fully connection neural network*.
f. *Import* MaxPooling2D dari *tensorflow keras* untuk operasi *pooling*. *Pooling layer* adalah lapisan proses reduksi data terjadi dengan pengecilan saja ukurannya.
g. *Import* ImageDataGenerator dari *tensorflow keras* untuk melakukan ekstraksi data yang berupa gambar atau citra digital, menjadi sebuah array yang dapat dibaca oleh TensorFlow.
h. *Import* numpy untuk mengaktifkan *library python*.

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data Augmentat
import numpy as numpy
# from livelossplot import PlotLossesKeras
```

```
len(tf.config.list_physical_devices('GPU')) > 0
```

```
True
```

Gambar 4.11 *Library* yang digunakan dan cek status GPU perangkat komputasi

Pada Gambar 4.12 merupakan proses pengaturan pengambilan data dari *storage* untuk proses *training* dan validasi. Pada proses ini hanya mengatur PATH yang sesuai dengan direktori pada *data train* dan *validate* sebelum memulai proses.

```
train = r'C:\Users\MSI-PC\Project\Driver-Drowsiness-Detection-using-Mobilenet\data\tra
```

```
test = r'C:\Users\MSI-PC\Project\Driver-Drowsiness-Detection-using-Mobilenet\data\test
```

Gambar 4.12 Pengaturan *storage* yang digunakan

Pada Gambar 4.13 *image_dataset_from_directory* merupakan sebuah *function* yang berfungsi untuk mengatur ukuran batch, ukuran gambar, parameter bersamaan dengan data pelatihan atau validasi pada data. Output dari *image_dataset_from_directory* adalah hasil kumpulan data dari dataset pelatihan, dan validasi. Pada tahap ini, *hyperparameter* diatur sebelum memulai proses *training*. *Batch size* diatur menjadi 32 agar sistem tidak terlalu berat melakukan *training,* artinya tiap epoch ada 268 iterasi karena jumlah data yang dilatih 8590 gambar. Dataset yang digunakan memiliki *image* berukuran 640x480 pixel, ini menjadi

ukuran yang sangat besar dan akan menyebabkan sistem terlalu berat untuk melakukan proses pelatihan, maka dilakukan *resize image* menjadi 224x224.

```
batchsize=32
```

```
from tensorflow.keras.preprocessing import image_dataset_from_directory
train_dataset = image_dataset_from_directory(train,
                                             shuffle=True,
                                             batch_size=batchsize,
                                             image_size=(224,224))
```

```
Found 67919 files belonging to 2 classes.
```

```
validation_dataset = image_dataset_from_directory(test,
                                                  shuffle=True,
                                                  batch_size=32,
                                                  image_size=(224,224))
```

```
Found 16979 files belonging to 2 classes.
```

Gambar 4.13 Pengaturan data sebelum pelatihan

Pada Gambar 4.14 dilakukan untuk melihat hasil pelabelan pada *data train.*

```
import matplotlib.pyplot as plt
class_names = train_dataset.class_names

plt.figure(figsize=(10, 10))
for images, labels in train_dataset.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Gambar 4.14 Plot label gambar *open* dan *closed*

Gambar 4.15 karena dataset asli tidak berisi set pengujian, tentukan berapa banyak kumpulan data yang tersedia di set validasi menggunakan tf.data.experimental.cardinality.

```
val_batches = tf.data.experimental.cardinality(validation_dataset)
test_dataset = validation_dataset.take(val_batches // 5)
validation_dataset = validation_dataset.skip(val_batches // 5)
```

```
print('Number of validation batches: %d' % tf.data.experimental.cardinality(validatior
print('Number of test batches: %d' % tf.data.experimental.cardinality(test_dataset))
```

```
Number of validation batches: 425
Number of test batches: 106
```

Gambar 4.15 Pembagian data

Selanjutnya menggunakan *buffered prefetching* untuk memuat gambar dari disk tanpa I/O menjadi *block* seperti pada Gambar 4.16.

```
AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

Gambar 4.16 Penggunaan *buffered prefetching*

Setelah itu, dilakukan augmentasi seperti pada Gambar 4.17. Teknik yang diterapkan adalah sebagai berikut:

- *RandomFlip* merupakan teknik augmentasi dengan membalikkan gambar secara acak selama proses *training,* dalam hal ini secara horizontal.
- *RandomRotation* merupakan teknik augmentasi dengan memutar posisi gambar secara acak sebesar 20 derajat.
- *RandomZoom* merupakan teknik augmentasi dengan memperbesar bagian gambar secara acak sebesar 20 persen.
- *RandomWidth* merupakan teknik augmentasi dengan mengubah lebar gambar 20 persen secara acak.
- *RandomHeight* merupakan teknik augmentasi dengan mengubah tinggi gambar 20 persen secara acak.

```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
  tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
  tf.keras.layers.experimental.preprocessing.RandomZoom(0.2),
  tf.keras.layers.experimental.preprocessing.RandomWidth(0.2),
  tf.keras.layers.experimental.preprocessing.RandomHeight(0.2),
])
```

Gambar 4.17 Teknik augmentasi gambar

Selanjutnya adalah membuat model dasar dari *pre-train* model yang digunakan seperti pada Gambar 4.18. Parameter yang dibutuhkan adalah sebagai berikut:

- *Input shape* adalah ukuran input yang telah di definisikan pada IMG_SHAPE.
- *Include top* diatur menjadi *false* karena bagian top klasifikasi akan dibuat sendiri.
- *Weights imagenet* adalah standar yang digunakan dalam klasifikasi gambar.

```
IMG_SHAPE = (224,224) + (3,)
base_model = tf.keras.applications.MobileNet(input_shape=IMG_SHAPE,
                                              include_top=False,
                                              weights='imagenet')
```

Gambar 4.18 Pengaturan *base model*

Tahap *modelling* dilakukan dengan menyatukan lapisan augmentasi yang telah dilakukan, penggunaan *base model* dan ekstraktor fitur menggunakan Keras seperti pada Gambar 4.19.

```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

Gambar 4.19 Pembuatan model dengan Keras

Pada Gambar 4.20 model di *compile* sebelum melatihnya karena ada dua kelas, tf.keras.losses.BinaryCrossentropy digunakan dengan *from_logits=False* karena model tidak memberikan output linier.

```
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate),
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
              metrics=['accuracy'])
```

Gambar 4.20 *Compile* model

Output yang ditampilkan saat menjalankan model.summary() seperti Gambar 4.15 terdapat 4.2 juta parameter di MobileNet dibekukan, tetapi ada 1.2 ribu parameter yang dapat dilatih. Hal ini dibagi antara dua tf.Variable objek, bobot dan bias.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
sequential (Sequential)      (None, None, None, 3)     0
_________________________________________________________________
MobilenetV3large (Functional (None, 7, 7, 1280)        4226432
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 1281
=================================================================
Total params: 4,227,713
Trainable params: 1,281
Non-trainable params: 4,226,432
_________________________________________________________________
```

Gambar 4.21 *Output model*

Setelah model berhasil di *compile*, maka selanjutnya adalah melakukan *training* seperti Gambar 4.22.

```
initial_epochs = 25

loss0, accuracy0 = model.evaluate(validation_dataset)
```

Gambar 4.22 *Training* dengan 25 epoch

Langkah selanjutnya adalah melakukan *un-freeze base_model* seperti pada Gambar 4.23 dan mengatur top layer agar tidak bisa dilatih.

```
base_model.trainable = True
```

Gambar 4.23 *Un-freeze base model*

Pada tahapan selanjutnya semua lapisan di *freeze* sebelum lapisan `fine_tune_at` seperti pada Gambar 4.24. Tujuan dilakukan *fine-tuning* adalah untuk mengadaptasi fitur-fitur khusus untuk bekerja dengan dataset baru.

```
base_model.trainable = True
```

```
# Let's take a look to see how many layers are in the base model
print("Number of layers in the base model: ", len(base_model.layers))

# Fine-tune from this layer onwards
fine_tune_at = 100

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
  layer.trainable =  False
```

```
Number of layers in the base model:  276
```

Gambar 4.24 Penggunaan *fine tuning*

Kemudian *compile* kembali model dengan yang menerapkan *fine tuning*. Penyesuaian bobot dan *learning rate* seperti Gambar 4.25 perlu dilakukan agar model tidak menjadi *over fit*.

```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
              optimizer = tf.keras.optimizers.RMSprop(lr=base_learning_rate/10),
              metrics=['accuracy'])
```

```
model.summary()
```

Gambar 4.25 Model di *compile* kembali

Setelah model di compile, dilanjutkan melatih model untuk meningkatkan akurasi dengan beberapa poin persentase seperti Gambar 4.26.

```
fine_tune_epochs = 25
total_epochs =  initial_epochs + fine_tune_epochs

history_fine = model.fit(train_dataset,
                         epochs=total_epochs,
                         initial_epoch=history.epoch[-1],
                         validation_data=validation_dataset)
```

Gambar 4.26 *Training* dengan 25 epoch tambahan *fine tuning*

Sekarang model telah siap untuk mengklasifikasikan mata tertutup dan mata terbuka untuk mendeteksi kelelahan berdasarkan kondisi mata. Pada tahap selanjutnya simpan model kedalam format h5 seperti Gambar 4.27.

```
# SAVE MODEL
model.save('model_mobilenetv1.h5')
```

Gambar 4.27 *Save model*

4.6 Source Code Confusion Matrix

Pada tahap pengujian model menggunakan *confusion matrix*, *library* yang digunakan dapat dilihat seperti pada Gambar 4.28. Adapun *library* yang dibutuhkan yakni:

i. *Import Sklearn* yaitu untuk mengaktifkan algoritma untuk bahasa python, penggunaannya untuk menampilkan akurasi.
j. *Import Pandas* yaitu *library* yang bersifat *open source*, menyediakan struktur data dan analisis data. Penggunaannya untuk membuat *data frame*.
k. *Import Matplotlib* yaitu untuk melakukan visualisasi data seperti membuat plot.
l. *Import Seaborn* yaitu *library* untuk visualisasi data, penggunaannya untuk membuat plot matrik konfusi untuk melihat prediksi mata tertutup dan mata terbuka.

```
import sklearn
import pandas
import matplotlib
import seaborn
```

Gambar 4.28 *Import library*

Langkah pertama adalah mendefinisikan data gambar beserta label dari *dataset valute* seperti Gambar 4.29. Kemudian terapkan model *sigmoid* pada model untuk mengembalikan *output* berupa *logit*. Jika *output* prediksi dan label berupa logit maka yang akan tampil berupa angka 0 atau 1.

```
#Retrieve a batch of images from the test set
image_batch, label_batch = validation_dataset.as_numpy_iterator().next()
predictions = model.predict_on_batch(image_batch).flatten()

# Apply a sigmoid since our model returns logits
predictions = tf.nn.sigmoid(predictions)
predictions = tf.where(predictions < 0.5, 0, 1)

print('Predictions:\n', predictions.numpy())
print('Labels:\n', label_batch)

plt.figure(figsize=(10, 10))
for i in range(9):
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(image_batch[i].astype("uint8"))
  plt.title(class_names[predictions[i]])
  plt.axis("off")
```

```
Predictions:
 [1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0]
Labels:
 [1 1 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0]
```

Gambar 4.29 Define data gambar dan *output logit*

Selanjutnya penerapan *true positive, false positive, true negative* dan *false negative* dengan kondisi jika prediksi sama dengan label maka hasil prediksi akan bertambah 1 seperti Gambar 4.30 dimana:

- True positive menampilkan jumlah data yang diprediksi 0 itu sesuai
- False positive menampilkan jumlah data yang diprediksi 1 itu sesuai
- True negative menampilkan jumlah data yang diprediksi 0 itu tidak sesuai
- False negative menampilkan jumlah data yang diprediksi 1 itu tidak sesuai

```
### True Positive
TP = 0

for i in range(0,len(label_batch)):
    if label_batch[i] == predictions.numpy()[i] and label_batch[i] == 1:
        TP+=1
print("True Positive: ", TP)
```

```
True Positive:  12
```

```
### False Positive
FP = 0

for i in range(0,len(label_batch)):
    if label_batch[i] == 0 and predictions.numpy()[i] == 1:
        FP+=1
print("False Positive: ", FP)
```

```
False Positive:  2
```

```
### True Negative
TN = 0
for i in range(0,len(label_batch)):
    if label_batch[i] == predictions.numpy()[i] and label_batch[i] == 0:
        TN+=1
print("True Negative: ", TN)
```

```
True Negative:  18
```

```
### False Negative
FN = 0
for i in range(0,len(label_batch)):
    if label_batch[i] == 1 and predictions.numpy()[i] == 0:
        FN+=1
print("False Negative: ", FN)
```

```
False Negative:  0
```

Gambar 4.30 Pencarian jumlah TP, FP, TN dan FN

Selanjutnya penerapan untuk mendapatkan hasil *correct prediction dan incorrect prediction* dengan kondisi jika prediksi sama dengan label maka hasil prediksi akan bertambah 1. *Correct prediction* akan menampilkan jumlah prediksi yang benar, sedangkan *incorrect prediction* menampilkan jumlah prediksi yang salah. Pengujian dilakukan dengan

32 data, sehingga jika nilai prediksi benar sejumlah 30 maka jumlah prediksi salah adalah 2, seperti pada Gambar 4.31.

```
### Correct Prediction
CP = 0
for i in range(0,len(label_batch)):
    if label_batch[i] == predictions.numpy()[i]:
        CP+=1
print("Correct Prediction: ", CP)
print(CP == TP + TN)
```

```
Correct Prediction:  30
True
```

```
### Incorect Prediction
ICP = 0
for i in range(0,len(label_batch)):
    if label_batch[i] != predictions.numpy()[i]:
        ICP+=1
print("Incorrect Prediction: ", ICP)
print(ICP == FP + FN)
```

```
Incorrect Prediction:  2
True
```

Gambar 4.31 Pencarian jumlah CP dan ICP

Pada Gambar 4.32 merupakan hasil *confussion matrix* untuk menghitung performa model. Hal ini ditujukan agar menjamin kualitas model sebelum di implementasikan.

```
### Confusion Matrik
image_batch, label_batch = validation_dataset.as_numpy_iterator().next()
predictions = model.predict_on_batch(image_batch).flatten()

# Apply a sigmoid since our model returns logits
predictions = tf.nn.sigmoid(predictions)
predictions = tf.where(predictions < 0.5, 0, 1)

print(sum([1 for e in label_batch if e ==1 ]))
print(sum([1 for e in label_batch if e ==0 ]))

print(sum([1 for e in predictions.numpy() if e ==1 ]))
print(sum([1 for e in predictions.numpy() if e ==0 ]))
```

```
17
15
18
14
```

Gambar 4.32 Evaluasi *confusion matrix*

Setelah melakukan evaluasi *confusion matrix,* selanjutnya adalah melihat seberapa baik model yang telah dibuat dari model yang dibuat menggunakan *metrics performance*

seperti yang dilakukan pada Gambar 4.33 dengan melihat nilai *precision, recall* dan *f1-score* sebagai berikut:

- nilai *precision* akan menunjukkan seberapa besar presisi atau akurasi model dalam prediksi
- nilai *recall* menunjukkan jumlah *class* yang ditangkap oleh model
- nilai F1 akan menjadi metrik yang digunakan ketika *precision* dan *recall* perlu diseimbangkan.

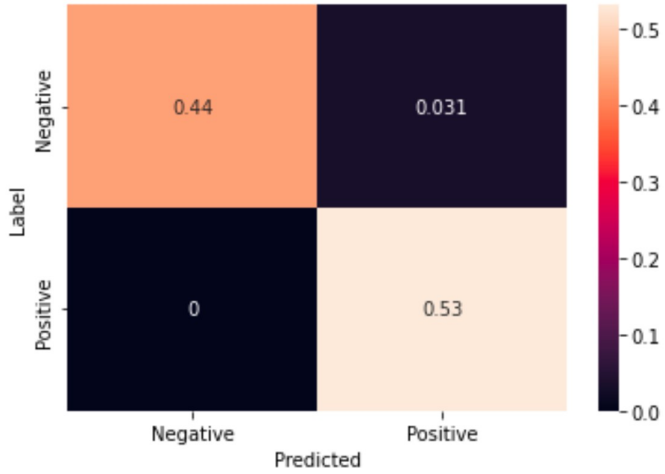
```
### matrik performa
from sklearn.metrics import classification_report
print(classification_report(label_batch,predictions.numpy()))
```

```
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        15
           1       0.94      1.00      0.97        17

    accuracy                           0.97        32
   macro avg       0.97      0.97      0.97        32
weighted avg       0.97      0.97      0.97        32
```

Gambar 4.33 *Metrics performance*

Selain itu, Gambar 4.34 menunjukan plot *heatmap* untuk melihat hasil evaluasi dengan lebih jelas.

```
### heatmap
import seaborn as sns
sns.heatmap(confusion , annot=True , xticklabels=['Negative' , 'Positive'] , yticklabe
plt.ylabel("Label")
plt.xlabel("Predicted")
plt.show()
```



Gambar 4.34 Menampilkan plot *heatmap*

BAB 5
Kesimpulan dan Saran

5.1 Kesimpulan

Setelah dilakukan pengujian terhadap model yang telah dibuat dengan menggunakan 20 persen jumlah dataset 84.900 dan dibagi menjadi *data train* sebanyak 67291, *data test* sebanyak 16979, serta dengan menggunakan 32 *validation data test* dapat ditarik kesimpulan sebagai berikut:

a. Model yang dihasilkan MobileNetV1 dipilih sebagai *best model* sesuai dengan tujuan karena dengan memiliki ukuran model terkecil yaitu 12 MB dibandingkan MobileNetV3 Large 32 MB dan EfficientNet Lite0 24 MB, akan tetapi tidak mengurangi akurasi prediksi sebesar 98% dengan *inference time* 0.05 detik.
b. Model berhasil di implementasikan dan di uji pada sistem pendeteksi kelelahan dengan mendeteksi kondisi mata subjek dan menghitung jumlah *score prediction* mata tertutup dan terbuka.

5.2 Saran

Keterbatasan *resource* komputasi untuk melakukan *training* dan evaluasi data membuat peneliti kurang banyak melakukan eksperimen, untuk pengembangan kedepannya bisa dilakukan dengan meningkatkan jumlah *batch size* 32 ke lebih tinggi dan data *test validati*on lebih banyak. Penelitian selanjutnya bisa membandingkan dengan arsitektur dengan domain yang berbeda.

Daftar Pustaka

UNICEF, “Indonesia: Survei Terbaru Menunjukkan Bagaimana Siswa Belajar Dari Rumah” https://www.unicef.org/id/press-releases/indonesia-survei-terbaru-bagaimana-siswa-belajar-dari-rumah (diakses 17 Agustus, 2019).

Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).

Suresh, Y., Khandelwal, R., Nikitha, M., Fayaz, M., & Soudhri, V. (2021, October). Driver Drowsiness Detection using Deep Learning. In 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC) (pp. 1526-1531). IEEE.

Adianto, F. K. (2021). Deteksi Kantuk Menggunakan Pendekatan Deep Learning Secara Real-time (Doctoral dissertation, Institut Teknologi Sepuluh Nopember).

Chmielińska, J., & Jakubowski, J. M. (2018). Detection of driver fatigue symptoms using transfer learning. Bulletin of the Polish Academy of Sciences, Technical Sciences, 66(6).

Feriawan, J., & Swanjaya, D. (2020, August). Perbandingan Arsitektur Visual Geometry Group dan MobileNet Pada Pengenalan Jenis Kayu. In Prosiding SEMNAS INOTEK (Seminar Nasional Inovasi Teknologi) (Vol. 4, No. 3, pp. 185-190).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

A. Santoso and G. Ariyanto, “Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah,” Emit. J. Tek. Elektro, vol. 18, no. 01, pp. 15–21, 2018, doi: 10.23917/emitor.v18i01.6235.

Ab Wahab, M. N., Nazir, A., Ren, A. T. Z., Noor, M. H. M., Akbar, M. F., & Mohamed, A. S. A. (2021). Efficientnet-lite and hybrid CNN-KNN implementation for facial expression recognition on raspberry pi. IEEE Access, 9, 134065-134080.

Khursheed, M. D., Khan, M. M., & Parveen, S. (2022). Real-time driver's drowsiness detection using transfer learning. In Application of Communication Computational Intelligence and Learning (pp. 73-82). Routledge.

Putra, I. W. S. E. (2016). Klasifikasi citra menggunakan convolutional neural network (CNN) pada caltech 101 (Doctoral dissertation, Institut Teknologi Sepuluh Nopember).

Wang, J., & Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit, 11, 1-8.

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

Goutte, C., & Gaussier, E. (2005, March). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In European conference on information retrieval (pp. 345-359). Springer, Berlin, Heidelberg.

Visa, S., Ramsay, B., Ralescu, A. L., & Van Der Knaap, E. (2011). Confusion matrix-based feature selection. MAICS, 710(1), 120-127.

Schwarz, C., Gaspar, J., Miller, T., & Yousefian, R. (2019). The detection of drowsiness using a driver monitoring system. Traffic injury prevention, 20(sup1), S157-S161.

Fudholi, D. H., Nayoan, R. A. N., Suyuti, M., & Rahmadi, R. (2021). Deteksi Indikasi Kelelahan Menggunakan Deep Learning. J-SAKTI (Jurnal Sains Komputer dan Informatika), 5(1), 1-9.

Susanto, P. E., Kurniawardhan, A., Fudholi, D. H., & Rahmadi, R. (2022). A Mobile Deep Learning Model on Covid-19 CT-Scan Classification. International Journal of Artificial Intelligence Research, 6(2).

Ahmad, R., & Borole, J. N. (2015). Drowsy driver identification using eye blink detection. IJISET-International Journal of Computer Science and Information Technologies, 6(1), 270-274.

Abtahi, S., Hariri, B., & Shirmohammadi, S. (2011, May). Driver drowsiness monitoring based on yawning detection. In 2011 IEEE International Instrumentation and Measurement Technology Conference (pp. 1-4). IEEE.

Rao, N. S., & Shetty, S. (2020). Drowsiness Detection System. International Journal of Research in Engineering, Science and Management, 3(05), 1-4.

Pheasant, S. (1991) Ergonomics, Work and Health, Macmillan Academic, Profesional Ltd, London

Caffier, P. P., Erdmann, U., & Ullsperger, P. (2005). The spontaneous eye-blink as sleepiness indicator in patients with obstructive sleep apnoea syndrome-a pilot study. Sleep Medicine, 6(2), 155-162.

Zhong, Z., Li, J., Luo, Z., & Chapman, M. (2017). Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework. IEEE Transactions on Geoscience and Remote Sensing, 56(2), 847–858.

Valueva, M. V., Nagornov, N.N., Lyakhov,P.A.,Valuev, G. V. Dan Chervyakov, N.I., 2020. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation, [daring] 177, hal.232–243.

LAMPIRAN A

Lampiran permintaan ijin penggunaan Dataset untuk eksperimen.

End User License Agreement

NTHU Drowsy Driver Detection (NTHU-DDD) Video Dataset

The video dataset consists of both male and female drivers, from different ethnicities, in 5 kinds of scenarios. The scenarios contain BareFace, Glasses, Sunglasses, Night-BareFace and Night-Glasses. Each frame in the video is labeled with either drowsy or non-drowsy state. The videos are captured under different situations, including day and night, with different types of drowsy and non-drowsy activities. The videos are in 640x480 pixels, 30 frames per second AVI format without audio.

By downloading the dataset you must agree to the following terms of use:

- Researcher shall use the database only for non-commercial research and educational purposes.
- The dataset cannot be distributed to second parties.
- Face images in the videos cannot be used for any other purposes, except that the face images for subject 010 & 021 can be included for academic publication use only.
- The following publication must be cited whenever making use of this dataset in any paper, publication, or report:
  Ching-Hua Weng, Ying-Hsiu Lai, Shang-Hong Lai, "Driver Drowsiness Detection via a Hierarchical Temporal Deep Belief Network", In Asian Conference on Computer Vision Workshop on Driver Drowsiness Detection from Video, Taipei, Taiwan, Nov. 2016
- National Tsing Hua University reserves the right to terminate researcher's access to the database at any time.

This agreement must be signed by the department head or lab director. (Please indicate in "Title".)

Ridho Rahmadi, Ph.D	ridho.rahmadi@uii.ac.id
Name	**Email Address**
Jl. Kaliurang KM.14, Sleman, D.I.Y. 55584	
Address	
	Department of Informatics Islamic University of Indonesia
Signature	**Institution**
Head of Research Laboratory	September 7, 2020
Title	**Date**