PENGARUH BIAS DATASET TERHADAP PERFORMA AKURASI DETEKSI OBJEK PADA ARSITEKTUR *EFFICIENTDET-LITE*



Disusun Oleh:

N a m a : Ridho Imantiyar

NIM : 18523170

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGARUH BIAS DATASET TERHADAP PERFORMA AKURASI DETEKSI OBJEK PADA ARSITEKTUR *EFFICIENTDET-LITE*

TUGAS AKHIR

Disusun Oleh:

N a m a : Ridho Imantiyar

NIM : 18523170

Yogyakarta, 25 Desember 2021

Pembimbing,

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

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGARUH BIAS DATASET TERHADAP PERFORMA AKURASI DETEKSI OBJEK PADA ARSITEKTUR *EFFICIENTDET-LITE*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 06 Januari 2022

Tim Penguji

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

Anggota 1

Dr. Syarif Hidayat, S.Kom., M.I.T. _____

Anggota 2

Irving Vitra Paputungan, S.T., M.Sc., Ph.D. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ridho Imantiyar

NIM : 18523170

Tugas akhir dengan judul:

PENGARUH BIAS DATASET TERHADAP PERFORMA AKURASI DETEKSI OBJEK PADA ARSITEKTUR *EFFICIENTDET-LITE*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 09 Februari 2022



(Ridho Imantiyar)

HALAMAN PERSEMBAHAN

Assalamualaikum Warahmatullahi Wabarakatuh

Alhamdulillahirobbil'alamin, segala puji syukur saya panjatkan kepada Allah, S.W.T. berkat izinnya, karunia, dan inayah-Nya yang telah memberikan saya kesehatan, kemudahan, kelancaran, serta keberkahan selama prsoses pengerjaan penelitian hingga proses penyelesaian tugas akhir ini. Dengan samapinya saya pada tahap ini, semoga bisa menjadi langkah yang baik dan memberikan dampak yang baik pula untuk masa depan saya.

Penyelesaian tugas akhir ini saya persembahkan kepada kedua orang tua saya, Bapak Badri dan Ibu Husnawati yang telah memberikan saya kesempatan untuk dapat menuntut ilmu di Universitas Islam Indonesia. Dan tak lupa pula ucapan terima kasih untuk segala dukungan dan doa yang selalu dipanjatkan disetiap ibadah yang dijalankan. Terima kasih juga untuk saudara-saudara saya dan seluruh keluarga yang telah memberikan dukungan secara langsung maupun tidak langsung.

Kepada Bapak Dhomas Hatta Fudholi, S.T, M.Eng, Ph.D., sebagai dosen pembimbing, saya ucapkan terima kasih atas segala arahan, usulan, dan bimbingannya sehingga saya dapat menyelesaikan tugas akhir ini.

Terima kasih saya ucapkan kepada semua rekan yang telah memberikan bantuan kepada saya untuk menyelesaikan laporan tugas akhir ini yang tidak bisa disebutkan satu per satu. Semoga Allah, S.W.T. selalu memberikan mereka kesehatan dan kemurahan rezeki.

Wassalamualaikum Warahmatullahi Wabarakatuh

HALAMAN MOTO

"Your time is limited, so don't waste it living someone else's life. Don't be trapped by dogma—which is living with the results of other people's thinking. Don't let the noise of others' opinions drown out your own inner voice. And most important, have the courage to follow your heart and intuition."

– Steve Jobs

KATA PENGANTAR

Segala puji dan syukur bagi Allah, S.W.T. atas berkat, rahmat, hidayah, serta karunianya sehingga penulis dapat menyelesaikan penyusunan Laporan Tugas Akhir dengan judul "PENGARUH BIAS DATASET TERHADAP PERFORMA AKURASI DETEKSI OBJEK PADA ARSITEEKTUR *EFFICIENTDET-LITE*" ini dengan bak. Tak lupa pula, sholawat serta salam penulis haturkan kepada tunjungan alam Nabi Muhammad SAW yang telah membimbing umatnya dari alam kegelapan menuju alam yang terang benderang.

Penyusunan laporan tugas akhir ini merupakan salah satu syarat yang harus dipenuhi untuk memperoleh gelar Sarjana Komputer pada Program Studio Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Selama proses penyusunan tugas akhir ini, penulis menyadari bahwa tanpa adanya dukungan, arahan, dan masukan dari berbagai pihak, penelitian ini tidak akan selesai dengan baik. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih kepada semua pihak yang telah berperan aktif dan penting dalam penyusunan laporan tugas akhir ini, terutama kepada:

1. Allah, S.W.T. yang telah memberikan kesehatan, rezeki, dan karunia-Nya sehingga penulis dapat menyusun dan menyelesaikan laporan tugas akhir dengan baik.
2. Kedua orang tua penulis, Bapak dan Ibu yang selalu mendo'akan serta memberikan perhatian dan kasih sayang kepada penulis serta memberikan motivasi dan dukungan agar penulis dapat menyelesaikan kuliah dengan baik.
3. Prof., Dr., Ir., Hari Purnomo M.T. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Hendrik, S.T., M.Eng. selaku Ketua Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., Ph.D. selaku Kaprodi Informatika, Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Dosen Pembimbing yang telah memberikan banyak saran, arahan dan masukan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan baik.
7. Seluruh Dosen dan Staf prodi Informatika. Terima kasih atas ilmu yang diberikan kepada penulis selama menuntut ilmu di Informatika.
8. Seluruh sahabat Roh Temaran yang selalu ada dalam sedih maupun duka dan memberikan semangat untuk bisa selalu bangkit.

9. Kepada Tim Hitrax Jakarta atas akses data yang bisa saya gunakan dalam penelitian ini.
10. Kepada Tim CDS UII yang telah melibatkan saya disuatu projek hingga mendapat inspirasi topik penelitian ini.
11. Kepada Yuyun, Luna, dan Fadhiila, yang telah banyak membantu dalam perantauan.
12. Kepada Hikmah Muliandari karena telah hadir dalam hidup penulis.
13. Semua pihak yang tidak bisa dituliskan satu persatu namun senantiasa mendukung baik secara langsung atau tidak langsung.

Penulis menyadari bahwa laporan tugas akhir ini masih memiliki banyak kekurangan dan jauh dari kata sempurna. Meskipun demikian, penulis berharap tulisan ini dapat memberikan manfaat kepada semua pihak. Oleh karena itu, penulis menerima segala bentuk kritik dan saran yang membangun dari pembaca untuk kebaikan bersama.

Yogyakarta, 25 Desember 2021



(Ridho Imantiyar)

SARI

Deteksi objek *(Object detection)* merupakan suatu metode yang dikembangkan untuk membuat sistem bisa mengenali suatu objek dalam foto atau video. Penelitian tentang deteksi objek telah banyak dilakukan oleh peneliti lainnya terutama dalam meningkatkan performa akurasi melalui konfigurasi model tetapi masih sedikit penelitian terkait peningkatan akurasi dari segi data. Pada data, terdapat bias yang memiliki pengaruh terhadap performa akurasi dekteksi objek akan tetapi penelitian terkait pengaruh bias terhadap performa model yang telah dikembangkan untuk perangkat bergerak masih minim dilakukan.

Pada penelitian ini akan digunakan model yang telah dioptimalkan untuk perangkat bergerak, yakni *efficientdet-lite* serta pengukuran performa model menggunakan nilai AP (*Average Precision*). AP merupakan salah satu matriks evaluasi yang digunakan untuk mengukur performa deteksi objek. Kemudian, penelitian ini akan menerapkan beberapa bias sebesar 50% terhadap data untuk mengukur dampak bias terhadap performa deteksi objek. Bias yang digunakan berupa bias pencahayaan, bias sudut pengambilan, dan bias pelabelan. Dari hasil penelitian yang sudah dilakukan, dataset yang memiliki bias masing-masing menghasilkan nilai AP sebesar 0,3918, 0,5215, dan 0,122. Sedangkan, dataset yang dijadikan sebagai nilai acuan menghasilkan nilai AP sebesar 0,6211.

Dalam penelitian ini, pengukuran pengaruh bias dilakukan dengan cara menghitung selisih antara nilai AP dataset bias dengan nilai AP dataset acuan. Dari perhitungan yang telah dilakukan, masing-masing bias memiliki selisih sebesar 36,9%, 16%, dan 98% dengan dataset acuan. Model dengan dataset bias pelabelan harus dihindari dan diminimalisir karena dapat menurunkan performa deteksi objek sebesar 98% sedangkan bias sudut pengambilan dan bias cahaya masih bisa dimanfaatkan untuk dijadikan dataset karena memiliki pengaruh yang masih kecil yakni sebesar 36,9% dan 16% dengan catatan jumlah data dengan bias tersebut harus dibawah 50% dari jumlah dataset.

Kata kunci: *dataset, bias dataset, efficientdet-lite*.

GLOSARIUM

Backbone	Sumber jaringan ekstraksi fitur yang digunakan dalam *deep learning*.
Machine Learning	ML merupakan bagian dari kecerdasan buatan yang menciptakan mesin untuk dapat dilatih berdasarkan data sehingga bisa memberikan suatu prediksi.
Ready to Drink	Salah satu jenis produk retail yang siap untuk diminum.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xi
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Batasan Masalah 4
1.4 Tujuan Penelitian 4
1.5 Manfaat Penelitian 4
1.6 Sistematika Penulisan 4
BAB II LANDASAN TEORI 6
2.1 *Artificial Intelligence* (Kecerdasan Buatan) 6
2.2 Machine Learning 7
2.3 Deep Learning 8
2.4 *Artificial Neural Network* 9
2.5 *Computer Vision* 10
2.6 Object Detection 10
2.7 Convolutional Network 11
2.7.1 Jenis Lapisan 12
2.7.2 *Dropout Regulation* 16
2.8 EfficientDet 16
2.8.1 Arsitektur 17
2.8.2 *Compound Scaling* 17

2.8.3 *EfficientDet-lite* 18
2.9 *Tensorflow* 19
2.10 Matrik Evaluasi 19
2.11 Bias Dataset 22
2.12 Studi Literatur 22
BAB III METODOLOGI PENELITIAN 27
3.1 Pengumpulan Data 27
3.1.1 Populasi dan Sampel 28
3.1.2 Jenis dan Sumber Data 29
3.2 Penyiapan Data 30
3.2.1 Praproses Gambar 31
3.2.2 Pembentukan Kondisi Dataset 31
3.2.3 Pelabelan Objek 37
3.3 Penyiapan Sistem Penelitian 40
3.4 Pemodelan 40
3.4.1 *Split Data*set 41
3.4.2 *Training* 41
3.4.3 Ekspor Model 43
3.5 Evaluasi Model 43
3.6 Menjalankan Model pada Aplikasi Perangkat Bergerak 44
3.7 Rancangan Antarmuka Aplikasi 45
3.8 Pengujian Model pada Perangkat Bergerak 46
BAB IV HASIL DAN PEMBAHASAN 48
4.1 Hasil Nilai *loss* 48
4.2 Hasil Skor Average Precision (AP) 48
4.3 Implementasi Model pada Perangkat Bergerak 50
4.4 Tampilan Antarmuka Aplikasi 53
4.5 Percobaan Model pada Perangkat Bergerak 55
BAB V PENUTUP 62
5.1 Kesimpulan 62
5.2 Saran 62
DAFTAR PUSTAKA 63
LAMPIRAN 65

DAFTAR TABEL

Tabel 2.1 Arsitektur penskalaan *EfficientDet* 18
Tabel 2.2 Arsitektur penskalaan *EfficientDet-lite* 19
Tabel 2.3 Perbandingan literatur tentang dataset deteksi objek 24
Tabel 2.4 Perbandingan literatur tentang bias data pada deteksi objek 25
Tabel 3.1 Nama kelas dan penampakan objek 29
Tabel 3.2 Jumlah gambar pada tiap jenis dataset 31
Tabel 3.3 Jenis dataset dan jenis bias yang diterapkan 32
Tabel 3.4 Jumlah label setiap kelas dan setiap dataset 40
Tabel 3.5 Hasil pengujian Tomas Repak 42
Tabel 3.6 Jumlah label pada data tes 44
Tabel 3.7 Contoh pengambilan gambar pada perangkat lunak 47
Tabel 4.1 Perbandingan *loss training* dan *loss validation* 48
Tabel 4.2 Perbandingan nilai AP model 48
Tabel 4.3 Perbandingan nilai AP tiap kelas 49
Tabel 4.4 Perbandingan hasil pengujian pada jarak 50 cm dan 100 cm 57
Tabel 4.5 Pengujian empat belas objek secara bersamaan 58
Tabel 4.6 Perbandingan jumlah label sebelum dan sesudah perbaikan pada dataset Optimal 59
Tabel 4.7 Nilai AP sebelum dan sesudah perbaikan pada model dataset Optimal 59
Tabel 4.8 Nilai AP setiap kelas sebelum dan sesudah perbaikan dataset Optimal 60
Tabel 4.9 Hasil Pengujian dengan dataset optimal sebelum dan sesudah diperbaiki 61

DAFTAR GAMBAR

Gambar 2.1 Bagian Utama dalam *Artificial Intelligence* (AI) 6
Gambar 2.2 Jaringan syaraf manusia 9
Gambar 2.3 Contoh arsitektur *Convolutional Neural Network* yang disederhanakan 12
Gambar 2.4 *Convolution Layer* 13
Gambar 2.5 Dua jenis *pooling* pada CNN 14
Gambar 2.6 Operasi *Max-Pooling* 14
Gambar 2.7 *fully-connected layer* 15
Gambar 2.8 *Dropout regulation* 16
Gambar 2.9 Arsitektur model *EfficientDet* 17
Gambar 2.10 Contoh kotak prediksi dan kota *ground truth* 20
Gambar 2.11 Ilustrasi perhitungan nilai IoU 21
Gambar 2.12 Penentuan TP, FP, dan FN 21
Gambar 3.1 Bagan alir metode penelitian 27
Gambar 3.2 Data produk *retail* 28
Gambar 3.3 Hasil pengambilan gambar 30
Gambar 3.4 Data gambar optimal 33
Gambar 3.5 Kondisi bias pertama 34
Gambar 3.6 Penurunan kecerahan citra 35
Gambar 3.7 Kondisi bias kedua 36
Gambar 3.8 Kondisi bias ketiga 37
Gambar 3.9 Pemberian label dengan *LabelImg* 38
Gambar 3.10 Isi file xml 39
Gambar 3.11 Baris kode persiapan *training* 42
Gambar 3.12 *Hyperparameter* model 43
Gambar 3.13 Baris Kode untuk menjalankan evaluasi 44
Gambar 3.14 Sub folder penyimpanan model pada projek aplikasi 45
Gambar 3.15 Tampilan rancangan aplikasi 46
Gambar 4.1 Kode untuk inisiasi pemanggilan fungsi memanggil gambar 50
Gambar 4.2 Fungsi untuk membuka kamera dan memproses gambar 51
Gambar 4.3 Fungsi untuk pendeteksian model dan menampilkan hasil deteksi 52
Gambar 4.4 Fungsi untuk pembuatan *bounding box,* nilai prediksi, dan label objek 53
Gambar 4.5 Baris kode untuk tampilan utama aplikasi 54

Gambar 4.6 Tampilan aplikasi deteksi objek.....55

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi, pada saat ini, telah menciptakan beragam solusi untuk memudahkan kehidupan manusia. Perkembangan tersebut tercipta karena adanya penelitian yang terus dilakukan oleh ilmuwan di seluruh dunia. Ranah penelitian yang dilakukan sangat beragam, salah satunya meneliti tentang fungsi dan cara berpikir manusia untuk diimplementasikan ke dalam suatu sistem yang mereka kembangkan. Hasilnya, sistem tersebut dapat memiliki kemampuan untuk mengenali suatu objek atau dapat disebut dengan *Object Detection*. *Object Detection* atau deteksi objek merupakan suatu metode yang dikembangkan untuk membuat sistem bisa mengenali suatu objek yang ada di dalam sebuah foto atau video. Secara umum, deteksi objek diciptakan untuk dapat mengidentifikasi objek seperti manusia, tumbuhan, kendaraan, atau objek lainnya yang sesuai dengan keinginan pengembang. Seiring dengan kemajuan teknologi, deteksi objek memiliki banyak penerapan di bidang visi komputer, termasuk dalam peningkatan automasi (Cahyo et al., 2020).

Secara umum, metode-metode komputasi dengan dasar pembelajaran mesin *(machine learning)* atau komputasi berdasar pembelajaran mendalam *(deep learning)* merupakan salah satu solusi atas permasalahan yang berkaitan dengan deteksi objek. Perbedaan kedua metode tersebut terletak pada cara belajar yang diterapkan. Pada *deep learning,* pembelajaran dilaksanakan secara mendalam dan kompleks. Salah satu algoritma *deep learning* yang kompleks dan menghasilkan akurasi yang baik dalam deteksi objek adalah *Convolutional Neural Network (CNN),* Mathworks (2017).

Convolutional Neural Network (CNN) merupakan algoritma yang dapat menerima masukan berupa gambar yang bisa digunakan oleh mesin untuk belajar mengenali masukan gambar tersebut. CNN merupakan dasar dari pengembangan metode deteksi objek. Salah satunya, yakni *TensorFlow Object Detection API*. Algoritma tersebut merupakan salah satu metode yang dapat diterapkan dalam mendeteksi objek. Metode tersebut menyediakan berbagai macam model pra-latih dan memiliki basis pembelajaran mendalam. *SSD Mobilenet V2* merupakan salah satu model yang umum digunakan dalam metode tersebut (Deng, 2019).

Kemudian, seiring dengan perkembangannya. Terdapat beberapa model CNN yang dikembangkan untuk dapat diaplikasikan ke dalam perangkat bergerak, yakni VGG16,

InceptionV3, ResNet50, MobileNet, dan Sqeezenet. Penerapan model CNN ke dalam perangkat bergerak memiliki beberapa keuntungan seperti menghasilkan *bandwith* komunikasi yang rendah terhadap server, biaya penggunaan *cloud computing* menjadi lebih murah, memperoleh waktu respon yang cepat, dan privasi data menjadi lebih terjaga. Selain itu, penerapan model ke dalam perangkat bergerak dapat membuat penggunaannya menjadi lebih mudah dan praktis. Namun, masih terdapat beberapa tantangan untuk dapat menghasilkan model yang memiliki waktu pemrosesan cepat, ukuran model kecil, serta tingkat akurasi yang tinggi. Salah satu contohnya, model ResNet50 memiliki ukuran sebesar 103 MB dan waktu pemrosesan 64 ms. Model tersebut masih memiliki kelemahan karena ukurannya yang masih besar (Deng, 2019). Belum lama ini, Tim Google Brain meluncurkan suatu model yang dapat mengatasi permasalahan tersebut. Melalui pustaka *TensorFlow Lite Model Maker,* lahirlah model pra-latih *EfficientDet-Lite* yang memiliki efisiensi dalam penerapannya, salah satu modelnya memiliki ukuran model sebesar 4,4 MB, waktu pemrosesan 37 ms, dan nilai *Average Precision* sebesar 25,69%. *EfficientDet-Lite* muncul dengan membawa teknik penskalaan yang baru dalam model CNN (Tan et al., 2020).

Selain menerapkan teknik komputasi, pemilihan metode, dan penggunaan model yang tepat, mengumpulkan data merupakan bagian yang penting juga. Tanpa keberadaan data maka sistem tidak dapat memulai proses pelatihan dan tentunya sistem tidak dapat mengenali objek. PASCAL VOC dan COCO merupakan dataset populer yang sering digunakan oleh peneliti untuk dijadikan sebagai tolak ukur dalam pengujian kemampuan model. (Shao et al., 2019). Saat ini, telah banyak tercipta kumpulan data yang memiliki skala besar dengan tujuan yang beragam. Data tersebut telah tersebar luas di internet serta dapat digunakan oleh pengembang untuk keperluan pengembangan sistem.

Hal penting terkait data, yakni data merupakan sumber pembelajaran mesin untuk dapat mengenali suatu objek sehingga hasil deteksi objek juga bergantung kepada data yang telah diolah. Meskipun data yang telah dikumpulkan banyak dan sudah merepresentasikan suatu objek, hal tersebut tidak dapat menjamin kemampuan deteksi yang dihasilkan akan baik juga. Dalam kumpulan data yang banyak pasti terdapat data yang kurang representatif di dalamnya atau dikenal dengan istilah bias. Selain itu, pasti akan ada kesalahan kecil yang ditimbulkan oleh manusia sehingga menambah bias dalam data (Khosla et al., 2012).

Bias yang terdapat dalam data merupakan masalah yang perlu diatasi untuk dapat menjaga serta meningkatkan kemampuan deteksi objek. Dalam penelitian yang dilakukan oleh Maeda et al dengan judul *"Road Damage Detection Using Deep Neural Networks with Images*

Captured Through a Smartphone" teknik pengambilan gambar yang ada pada dataset berbeda dengan teknik pengambilan gambar pada saat pengujian. Kejadian tersebut mengakibatkan sistem tidak dapat melakukan pendeteksian. Bias yang terjadi pada penelitian tersebut merupakan bias pengambilan gambar. Jenis-jenis bias yang umum terjadi pada kumpulan data gambar telah dikelompokkan oleh Tommasi et al pada penelitiannya yang berjudul *"A Deeper Look at Datset Bias".*

Berdasarkan uraian di atas, penerapan model deteksi objek ke dalam perangkat bergerak memiliki keuntungan seperti, lebih mudah untuk digunakan, menghemat biaya komputasi awan, dan mendapat respon yang cepat. Kemudian, data memiliki peran penting terhadap performa akurasi deteksi objek dalam mengenali suatu objek yang ada di dalam gambar. Selain itu, di dalam data pun terdapat bias data yang dapat mempengaruhi performa akurasi deteksi objek. Oleh karena itu, penelitian ini dilakukan dengan tujuan untuk mengukur pengaruh bias dataset terhadap performa akurasi model deteksi objek berdasarkan nilai AP yang dihasilkan oleh model tersebut kemudian model tersebut akan diimplementasikan dalam perangkat bergerak berbasis android. Bias data yang terdapat pada datset dipengaruhi oleh beberapa aspek, yakni teknik pengambilan gambar, teknik anotasi data, dan teknik dalam mengatasi bias yang ada di dalam kumpulan data (Tommasi et al., 2017).

Pada penelitian ini, peneliti akan membuat empat jenis dataset yang berbeda, satu dataset tanpa bias yang akan dijadikan sebagai acuan untuk mengukur pengaruh bias dan tiga dataset yang memiliki bias. Bias yang diterapkan berupa bias arah datangnya cahaya, bias pengambilan gambar, dan bias pelabelan objek. Kemudian, keempat dataset tersebut akan dilatih menggunakan model *EfficentDet-Lite* yang telah dikembangkan melalui pustaka *TensorFlow Lite Model Maker.* Penggunaan model tersebut dikarenakan model tersebut telah dioptimalkan untuk dapat berjalan di perangkat bergerak yang memiliki luaran model dengan ukuran sebesar 4,4 MB dan latensi terendah yang dihasilkan sebesar 37ms (Tan et al., 2020). Selesai pelatihan, keempat dataset tersebut akan menghasilkan empat buah model. Kemampuan model diukur menggunakan nilai AP. Kemudian, pengukuran bias dilakukan dengan cara menghitung selisih nilai AP antara model yang menggunakan dataset bias dengan model yang menggunakan dataset tanpa bias. Hasil dari penelitian diharapkan dapat menjadi rujukan dalam pembuatan dataset yang sesuai dengan kebutuhan pengembang atau penelitian sehingga menghasilkan performa akurasi deteksi objek yang baik dan sesuai dengan target.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, rumusan masalah yang akan diteliti adalah bagaimana pengaruh bias data terhadap performa model deteksi objek pada arsitektur perangkat bergerak dengan menggunakan model *EfficientDet-Lite*?

1.3 Batasan Masalah

Untuk menjaga fokus dalam penelitian ini agar tidak menyimpang, berikut batasan yang perlu diperhatikan;

a. Model yang akan digunakan dalam penelitian akan terfokus pada *EfficientDet-Lite*
b. Objek dalam penelitian ini berasal dari produk retail khususnya produk dalam kategori RTD *(ready to drink)*.
c. Jumlah kelas yang diugunakan dalam pelatihan data sebanyak empat belas kelas.
d. Performa model dilihat dari nilai evaluasi yang dihasilkan oleh matriks *Average Precision* (AP)

1.4 Tujuan Penelitian

Sesuai dengan latar belakang dan rumusan masalah, penelitian ini bertujuan untuk mengukur besar pengaruh bias dataset terhadap performa akurasi deteksi objek serta melakukan implementasi model pada perangkat bergerak berbasis android. Pengukuran bias dilakukan dengan cara menghitung besar selisih nilai AP antara model yang dijadikan acuan dengan model yang memiliki bias data.

1.5 Manfaat Penelitian

a. Membantu data scientist untuk memberikan gambaran dari pengaruh bias dataset yang telah dianalisis oleh peneliti.
b. Penelitian ini dapat menjadi rujukan bagi peneliti lain yang akan mengimplementasikan *deep learning* ke dalam perangkat bergerak.
c. Menjadi solusi baru dalam meningkatkan performa akurasi deteksi objek dengan cara memaksimalkan kualitas dataset.

1.6 Sistematika Penulisan

Sistematika penulisan laporan pada tugas akhir akan dibagi ke dalam lima bagian, yakni:

BAB I PENDAHULUAN

Dalam bab ini membahas masalah yang terkait dengan latar belakang penelitian, rumusan masalah, Batasan masalah, tujuan penelitian, dan manfaat penelitian yang merupakan gambaran secara umum dari penelitian ini.

BAB II LANDASAN TEORI

Dalam bab ini membahas tentang teori yang mendasari analisis permasalahan yang memiliki hubungan dengan pembahasan.

BAB III METODE PENELITIAN

Dalam bab ini, membahas tentang metode yang digunakan penulis dalam pengumpulan data, pemrosesan data, perancangan sistem, serta evaluasi hasil yang dilakukan dalam penelitian

BAB IV HASIL DAN PEMBAHASAN

Dalam bab ini membahas tentang hasil dan pembahasan dari rancangan data yang telah dilatih menggunakan model *efficientdet-lite2*.

BAB V PENUTUP

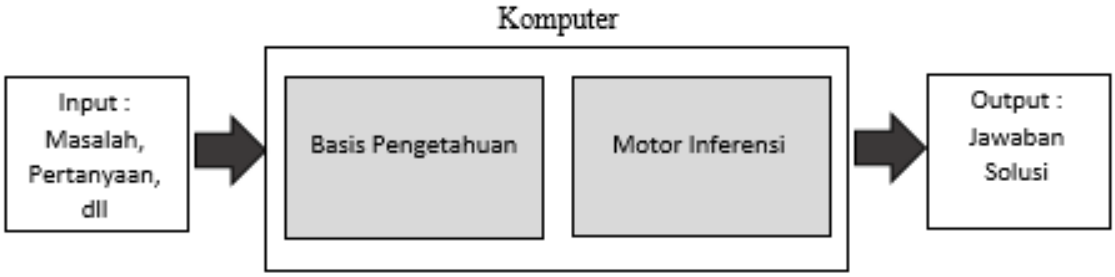
Dalam bab ini berisi kesimpulan dari semua proses yang telah dilakukan dalam penelitian ini serta memberikan saran yang diharapkan dapat dimanfaatkan untuk penelitian yang dilakukan di waktu yang akan mendatang.

BAB II
LANDASAN TEORI

2.1 *Artificial Intelligence* (Kecerdasan Buatan)

Artificial Intelligence merupakan suatu sistem yang melakukan proses simulasi dari kecerdasan manusia yang dilakukan oleh mesin, terutama sistem komputer. *Artificial Intelligence* berfokus pada perolehan data dan pembuatan aturan tentang cara mengubah data menjadi informasi yang dapat ditindaklanjuti. *Aritifial Intelligence* (AI) merupakan bagian dari ilmu komputer yang mempelajari cara menjadikan komputer untuk dapat melakukan pekerjaan yang dapat meniru manusia dan bahkan bisa melakukan pekerjaan sebaik yang dilakukan oleh manusia. *Artificial Intelligence* (AI) menurut John McCarthy (1956) yang dikutip dari jurnal penelitian (Pannu, 2015) menyebutkan bahwa AI memiliki tujuan untuk mengetahui atau memodelkan proses berpikir manusia dan merancang sistem hingga dapat meniru prilaku manusia. Sistem ini tidak harus memiliki batasan terhadap metode yang diamati secara biologis (McCarthy, 2007). Dalam pembuatan sistem AI terdapat dua hal utama yang paling dibutuhkan, yakni :

a. Knowledge Base (Basis Pengetahuan). Bagian ini berisi tentang fakta-fakta, teori, pemikiran, dan relasi antar satu dengan yang lainnya.
b. *Inference Engine* (Motor Inferensi), yaitu kemampuan untuk menyimpulkan suatu hal yang berdasarkan kepada pengalaman.



Gambar 2.1 Bagian Utama dalam *Artificial Intelligence* (AI)

Artificial Intelligence (AI) adalah salah satu disiplin keilmuan yang amat luas, sebgian lingkup utama AI adalah Sistem Pakar (*Expert System*), Pengolahan Bahasa Alami (*Natural Language Processing/NLP*), Pengenalan Ucapan (*Speech Recognition*), *Computer Vision, Intelligent Computer-Aided Instruction,* dan lainnya. Siste, pakar merupakan kemampuan yang

dapat untuk menirukan seorang pakar. Maksud dari sistem tersebut untuk dapat melakukan transfer kepakaran dari seorang pakar ke komputer, kemudian ke orang lain (orang yang bukan pakar). Pengolahan Bahasa Alami merupakan kegiatan komunikasi yang dilakukan antara komputer dengan manusia. Komunikasi tersebut menggunakan bahasa sehari-hari yang biasa digunakan oleh manusia. Kemudian, pengenalan ucapan dapat diartikan sebagai proses komunikasi yang dilakukan manusia dengan komputer menggunakan media suara. Selanjutnya, *Computer Vision* adalah proses menginterpretasikan objek atau gambar yang tampak melalui komputer. *Intelligent Computer-Aided Instruction* adalah cara komputer untuk dapat berberap sebagai pengarah yang bisa mengajar atau melatih.

Artificial Intelligence (AI) dibuat berdasarkan sistem yang mempunyai kemampuan seperti manusia pada domain tertentu yang dikenal dengan istilah *soft computing. Soft computing* adalah inovasi baru dalam menciptakan sistem cerdas yang bisa melakukan adaptasi dan melakukan pekerjaan yang lebih baik jika terjadi perubahan pada lingkungan. *Soft computing* juga dapat melakukan eksploitasi terhadap adanya toleransi terhadap ketidakpastian, ketidaktepatan, dan kebenaran parsial sehingga bisa menyelesaikan dan mengendalikan lingkungan agar sesuai dengan realita. Metodelogi yang kerap diaplikasikan dalam *soft computing* salah satunya, yakni Jaringan Saraf Tiruan (*Artificial Neural Network/ANN*). Metodelogi lain yang juga dimanfaatkan adalah Sistem *Fuzzy* (mengakomodasi ketepatan), *Probabilistic Reasoning* (Mengakomodasi Ketidakpastian), *Evolutionary Computing* (Optimasi).

Menurut Rich dan Knight (1991), AI merupakan sebuah studi tentang cara untuk dapat membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia. Pada awal pembuatannya, komputer hanya digunakan sebagai alat untuk melakukan perhitungan. Akan tetapi, seiring dengan berkembangnya zaman ini, komputer tidak hanya dipergunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk menyelesaikan segala macam masalah yang bisa mempercepat pekerjaan manusia.

2.2 Machine Learning

Sebutan *machine learning* muncul pertama kali oleh Arthur Samuel pada tahun 1959. Menurutnya, *machine learning* merupakan salah satu bidang ilmu komputer yang memberikan kemampuan pembelajaran kepada komputer untuk mengetahui sesuatu tanpa pemrograman yang jelas. Menurut Mohri dkk (2012) *machine learning* dapat diartikan sebagai suatu langkah

komputasi berdasarkan pengalaman untuk meningkatkan performa atau membuat prediksi yang akurat. Definisi pengalaman disini merupakan informasi sebelumnya yang telah tersedia dan bisa dijadikan data pembelajar.

Dalam pembelajaran *machine learning,* terdapat beberapa scenario-skenario seperti:

a. *Supervised Learning*

Penggunaan scenario supervised learning, pembelajaran menggunakan masukan data pembelajran yang telah diberi label. Setelah itu membuat prediksi dari data yang telah diberi label.

b. *Unsupervised Learning*

Penggunaan scenario Unsupervised Learning, pembelajaran menggunakan masukan data pembelajaran yang tidak diberi label. Setelah itu mencoba untuk mengelompokan data berdasarkan karakterisitk-karakteristik yang ditemui.

c. *Reinforcement Learning*

Pada scenario Reinforcement Learning, fase pembelajaran dan tes saling dicampur. Untuk mengumpulkan informasi pembelajar secara aktif dengan berinteraksi ke lingkungan sehingga untuk mendapatkan balasan untuk setiap aksi dari pembelajar.

Saat ini telah banyak pendekatan *machine learning* yang diaplikasikan untuk deteksi *spam, Optical character recognition* (OCR), pengenalan wajah, deteksi penipuan *online,* NER (*Named Entity Recognition*), *Part-of-Speech Tagger.*

2.3 Deep Learning

Deep learning adalah salah satu bidang dari *Machine Learning* yang menggunakan jaringan syaraf tiruan untuk aplikasi permasalahan dengan dataset yang besar. Teknik *Deep Learning* dapat membangun arsitektur yang amat kuat untuk *Supervised Learning*. Penambahan lapisan yang lebih banyak dapat membuat model pembelajaran bisa mewakili data citra yang menggunakan label degan lebih bagus. Pada *Machine Learning,* terdapat teknik untuk memakai ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk melakukan klasifikasi citra mauoun mengenali suara. Namun, metode tersebut masih memiliki beberapa kekurangan baik dalam hal kecepatan dan akurasi.

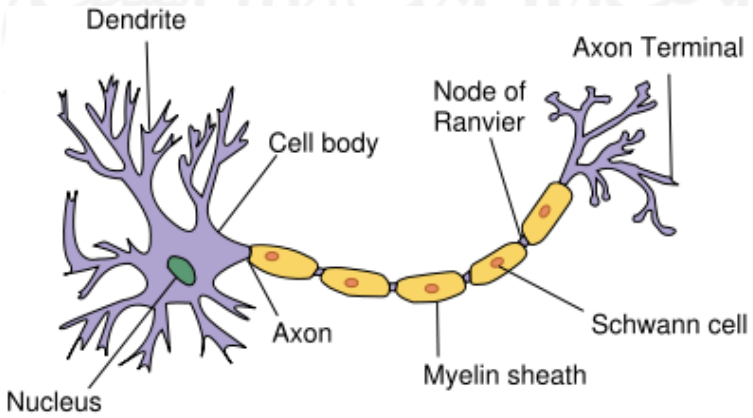
Aplikasi konsep jaringan saraf tiruan yang dalam (banyak lapisan) dapat ditangguhkan pada algoritma *Machine Learning* yang sudah ada sehingga komputer sekarang bisa belajar dengan kecepatan, akurasi, dan skala yang besar. Prinsip ini terus berkembang hingga *Deep Learning* semakin sering digunakan pada komunitas riset dan industry untuk membantu

memecahkan banyak masalah data besar seperti *Computer vision, Speech recognition,* dan *Natural Language Processing. Feature Engineering* adalah salah satu fitur utama dari *Deep Learning* untuk melakukan ekstraksi pola yang dapat digunakan untuk memudahkan model membedakan kelas. *Feature Engineering* juga dapat diartikan sebagai teknik yang sangat penting meraih hasil yang baik pada saat melakukan prediksi. Namun, hal tersebut masih sulit untuk dapat dipelajari dan dikuasai karena kumpulan data dan jenis data yang berbeda memerlukan pendekatan teknik yang berbeda juga.

Algoritma yang digunakan pada *Feature Engineering* dapat mengidentifikasi pola umum yang penting untuk membedakan antara kelas dalam *Deep Learning,* motode CNN atau *Convolutional Neural Network* sangatlah bagus dalam menemukan fitur yang baik pada citra ke lapisan berikutnya untuk membentuk jipotesis nonlinier yang dapat meningkatkan sebuah model. Model yang kompleks tentunya akan mebutuhkan waktu pelatihan yang lama sehingga di dunia *Deep Learning* penggunaan GPU sudah sangatlah umum (Munir, 2004).

2.4 *Artificial Neural Network*

Artificial Neural Network (ANN) adalah sebuah model komputasi parallel yang meniru fungsi dari sistem jaringan syaraf biologi otak manusia. ANN merupakan sistem adatif yang dapat mengubah strukturnya untuk memecahkan suatu masalah berdasarkan informasi *internal* maupun *eksternal* (Pham, 1994). Dalam Otak manusia, neuron memiliki jumlah dari ribuan hingga milyaran neuron yang saling berhubungan. Hubungan ini disebut dengan *Synapses*. Komponen neurin terdiri dari satu inti sel yang akan mengerjakan pemrosesan informasi, satu akson (*axon*) dan minimal satu *dendrite.* Informasi yang masuk akan diterima oleh *dendrite.* Selain itu, *dendrite* juga menyertasi akson sebagai keluaran dari suatu pemrosesan informasi.



Gambar 2.2 Jaringan syaraf manusia

Cara kerja dari sistem syaraf pada Gambar 2.2 adalah berawal dari sinyal yang masuk dari *dendrite* kemudian bergerak menuju *cell body*. Kemudian sinyal tersebut akan diproses di dalam badan sel berdasarkan fungsi tertentu (*Summation Proses*). Jika sinyal hasil proses melebihi nilai ambang batas (*threshold*) tertentu maka sinyal tersebut akan membangkitkan neuron untuk meneruskan sinyal tersebut. Sedangkan, jika dibawah nilai ambang batasnya maka sinyal tersebut akan dihalangi (*inhibited*). Kemudian, sinyal yang diteruskan akan menuju ke *axon* dan akhirnya menuju ke neuron lain yang melewati *synapse.*

2.5 *Computer Vision*

Secara sederhana, *computer vision* merupakan cara komputer atau mesin dapat melihat objek. *Computer vision* juga dapat diartikan sebagai bidang yang melingkupi metode untuk memperoleh, menganalisis, mengolah, dan memahami data visual seperti gambar atau video. Dalam c*omputer vision,* terjadi sebuah proses transformasi data yang bersumber dari kamera video maupun foto atau gambar ke dalam suatu hasil keputusan ataupun sebuah presentasi yang baru, dimana hasil dari proses transformasi tersebut memiliki kepentingan untuk mencapai sebuah tujuan (Cahyo et al., 2020). *Computer vision* memiliki tujuan utama agar komputer atau mesin dapat meniru hingga mengungguli kemampuan perseptual mata manusia dan otak sehingga *computer vision* dapat menjadikan komputer "*acts like human sight*". Kemampuan tersebut merupakan kombinasi antara pengolahan citra (*image processing*) dan pengolahan pola (*pattern recognition*).

Pengolahan citra merupakan bidang yang berkaitan dengan proses perubahan citra gambar. Proses ini memiliki tujuan untuk memperoleh kualitas citra yang lebih baik. Sedangkan Pengenalan pola adalah bidang yang berhubungan dengan proses identifikasi citra yang memiliki tujuan untuk mengekstrak sejumlah informasi yang disampaikan oleh gambar.

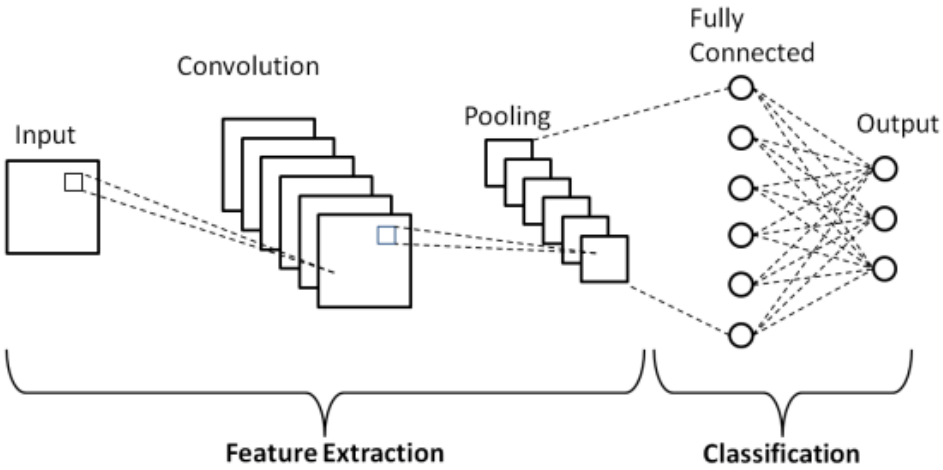
2.6 Object Detection

Object detection atau dalam bahasa Indonesia disebut dengan deteksi objek merupakan sistem yang dapat mengenali suatu objek yang ada di suatu gambar atau video. Deteksi objek memiliki banyak penerapan dalam visi komputer termasuk dalam deteksi wajah, melacak pergerakan orang dalam video, atau melacak objek lainnya. Bidang deteksi objek secara umum melakukan pencarian di setiap bagian gambar untuk melokalisasi bagian yang memiliki kesamaan dengan dataset pelatihan. Ketika gambar yang dipindai memiliki kesamaan atau kemiripan dengan dataset pelatihan maka sistem dapat memberikan suatu luaran sesuai dengan

nama objek yang berhasil dipindai. Kesamaan antara masukan gambar dan hasil pelatihan dataset dapat diukur dengan korelasi. Selama beberapa tahun ini, deteksi objek berbasis gambar telah menunjukkan hasil yang bergantung terhadap data pelatihan (Harani & Hasanah, 2020).

2.7 Convolutional Network

Convolutional Neural Network yang biasa disebut dengan CNN adalah pengembangan dari *multilayer perceptron* (MLP) yang dibuat untuk memproses data dua dimensi dalam bentuk gambar. CNN termasuk kedalam jenis *Neural Network* dikarenakan memiliki tingkat kedalaman jaringan yang tinggi serta penerapannya banyak dilakukan pada data gambar. Pada dasarnya, pengklasifikasian citra dapat menggunakan MLP, akan tetapi dengan metode tersebut terdapat ketidaksesuaian karena meotode tersebut tidak menyimpan informasi khusus dari data gambar dan menganggap setiap piksel adalah fitur yang berdiri sendiri sehingga menghasilkan hasil yang kurang baik. Penelitian awal yang mendasari penemuan CNN ini pertama kali dilaksanakan oleh Hubel dan Wiesel (Hubel & Wiesel, T, 1968) tentang *viual cortex* pada indera penglihatan kucing. Secara teknis, CNN merupakan suatu arsitektur yang dapat dilatih dan terdiri dari beberapa tahap. *Input* dan *Output* dari setiap proses terdiri dari beberapa *array* yang biasa disebut dengan *feature map.* Pada dasarnya, CNN merupakan suatu arsitektur yang bisa dilatih dengan tahapan yang dimilikinya. Di dalam tahapan tersebut terdapat beberapa lapisan, yakni lapisan konvolusi, lapisan *pooling,* dan lapisan *fully-connected*. Gambar 2.3 merupakan arsitektur dari *Convolutional Neural Network* yang telah disederhanakan:



Gambar 2.3 Contoh arsitektur *Convolutional Neural Network* yang disederhanakan
Sumber: Phung & Rhee (2019)

Berdasarkan gambar diatas, CNN pertama kali melakukan tahap konvolusi. Pada tahap tersebut, terdapat penggunaan suatu kernel yang memiliki ukuran tertentu. Banyaknya kernel yang digunakan dihitung berdasarkan dari jumlah fitur yang diperoleh. Selanjutnya diteruskan menuju fungsi aktivasi, umumnya memakai fungsi aktivasi ReLU (*Rectifier Linear Unit)*. Berikutnya, setelah proses fungsi aktivasi selesai maka akan dilanjutkan menuju tahap *pooling*. Langkah-langkah tersebut diulang beberapa kali hingga memperoleh peta fitur yang cukup untuk diteruskan menuju *fully connected network* kemudian berakhir pada *output.*

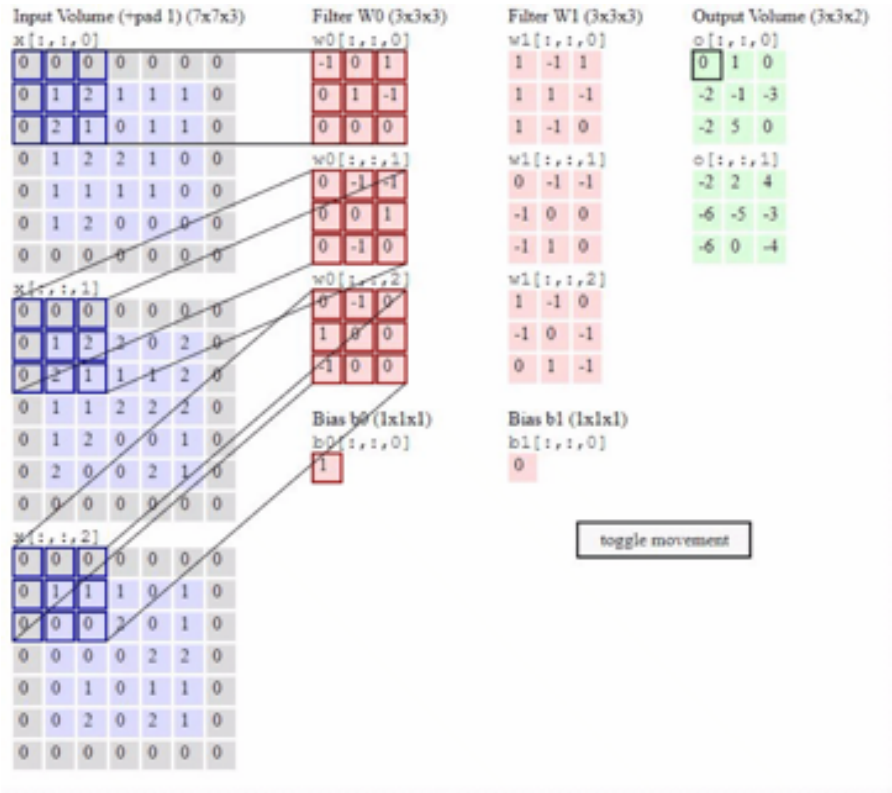
2.7.1 Jenis Lapisan

Lapisan Konvolusi

Pada tahap ini dilakukan proses konvolusi dengan suatu filter ketika memindai masukan yang berkaitan dengan dimensinya. Konvolusi merupakan sebutan matematis dalam penerapan suatu fungsi pada luaran fungsi lain secara berulang. Operasi konvolusi adalah operasi yang memiliki nilai nyata dalam dua fungsi argumen. Operasi tersebut mengaplikasikan fungsi luaran sebagai *feature map* dari masukan gambar. Masukan dan luaran tersebut dapat dimaksud sebagai dua argumen yang memiliki nilai nyata. *Hyperparameter* dari operasi konvolusi mencakup ukuran filter F dan langkah S. Operasi ini bertujuan untuk mengekstrak *high-level features* berupa pinggiran, warna, orientasi gradien, dsb.

Filter dengan panjang dan tinggi (*pixels*) yang terbentuk di dalam *convolution layer* terdiri dari *neuron*. Contohnya, lapisan pertama pada *feature extraction layer* umumnya merupakan *convolutional layer* yang memiliki ukuran 7x7x3. Panjang 7 *pixels*, tinggi 7 *pixels*,

dan tebal/jumlah 3 buah sesuai dengan *channel* dari gambar. Ketiga filter tersebut akan ditranslasi ke semua area dari gambar. Setiap perpindahan akan dilakukan operasi "dot" antara masukan dan nilai dari filter tersebut. Dari proses tersebut akan dihasilkan suatu luaran yang biasa diistilahkan sebagai *activation map* atau *feature map*. Lihatlah ilustrasi dari Gambar 2.4 untuk bisa melihat proses yang terjadi pada *convolution layer.*

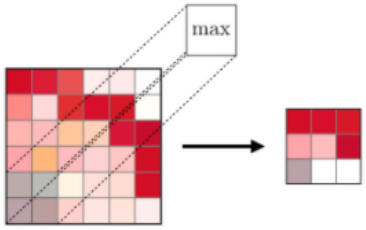
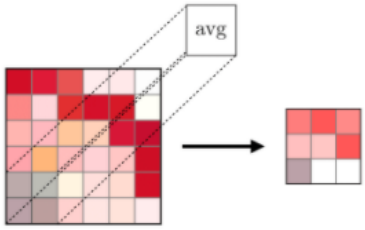


Gambar 2.4 *Convolution Layer*

Sumber : Medium Samuel Sena (2017)

Pooling

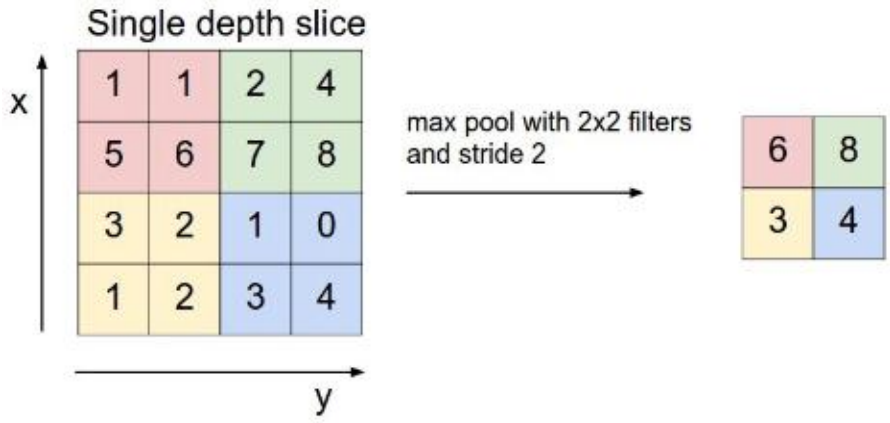
Pooling adalah proses yang dilakukan untuk mengurangi ukuran matriks dengan menerapkan operasi *pooling* untuk tahap berikutnya. Selain itu juga memiliki manfaat untuk mengekstrak *feature dominant* yang melakukan beberapa invariansi spasial. *Pooling layer* memiliki dua macam jenis yang biasa digunakan, yakni *max pooling* dan *average pooling*. Penjelasan dari kedua jenis tersebut dapat dilihat pada Gambar 2.5.

Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

Gambar 2.5 Dua jenis *pooling* pada CNN

Sumber: Afshine (2019)

Pada dasarnya, lapisan ini terdiri dari sebuah filter yang memiliki ukuran dan stride tertentu yang akan secara bergantian bergeser pada seluruh area *feature map.* Dalam *poolinglayer*, nilai yang diambil pada *average pooling* adalah nilai rata-rata, sedangkan pada *max-pooling* mengambil nilai maksimal. Lapisan ini dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model CCN sehingga model tersebut dapat secara progresif mengurangi ukuran volume luaran pada *Feature Map*. Pengurangan tersebut dapat mengendalikan *Overfitting.* Lapisan *pooling* bekerja pada setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk lapisan *pooling* umumnya menggunakan filter dengan ukuran 2x2 yang diterapkan dengan tahapan sebanyak dua dan bekerja pada setiap irisan masukannya.



Gambar 2.6 Operasi *Max-Pooling*

Sumber: Medium Samuel Sena (2017)

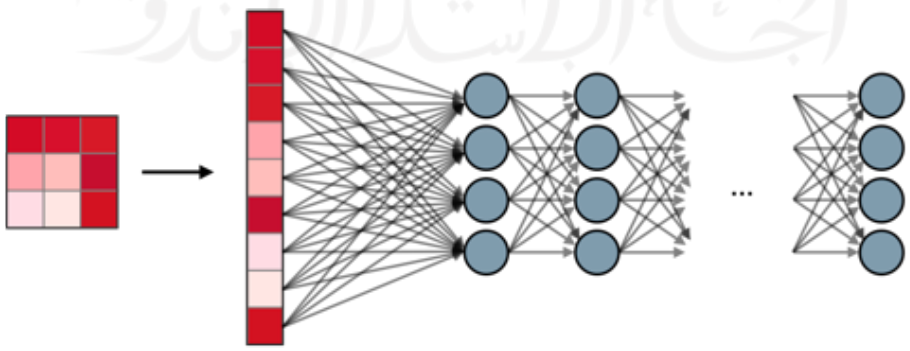
Gambar 2.6 menunjukkan proses dari *max-pooling.* Luaran dari tahapan *pooling* adalah sebuah matriks dengan dimensi yang lebih kecil dari dimensi citra awal. Lapisan *pooling* pada Gambar 2.6 beroperasi pada setiap irisan kedalaman volume masukan dengan cara bergantian. Jika dilihat dari gambar operasi *max-pooling* dengan menggunakan ukuran filter 2x2. Masukan pada proses tersebut memiliki ukuran 4x4, dari masing-masing 4 angka pada input operasi tersebut diambil nilai tertingginya kemudian dilanjutkan membuat ukuran luaran baru menjadi ukuran 2x2.

Fully-Connected Layer

Suatu lapisan yang menjadi tempat untuk neuron aktivasi yang berasal dari lapisan sebelumnya untuk terhubung dengan semua neuron yang berasal dari lapisan selanjutnya. Pada dasarnya, lapisan ini umumnya biasa digunakan pada MLP (Multi Layer Perceptron) yang memiliki maksud untuk melakukan transformasi di dimensi data supaya data bisa diklasifikasikan secara linear. Secara umum, lapisan ini bertujuan untuk mentransformasi dimensi data agar bisa diklasifikasikan dengan cara linear. Kemudian, lapisan ini berada pada tahap akhir arsitektur CNN dan dapat dimanfaatkan untuk mengoptimalkan tujuan seperti nilai kelas. Berikut merupakan ilustrasi dari *fully-connected* layer yang dapat dilihat pada

Gambar 2.7.

Perbedaan antara lapisan *fully-connected* dan lapisan konvolusi biasa terletak pada neuron di lapisan konvolusi yang terhubung hanya ke daerah tertentu pada input, sementara lapisan *fully-connected* mempunyai neuron yang secara keseluruhan terhubung akan tetapi kedua lapisan tersebut masih melakukan operasi produk dot sehingga fungsinya tidak begitu berbeda.

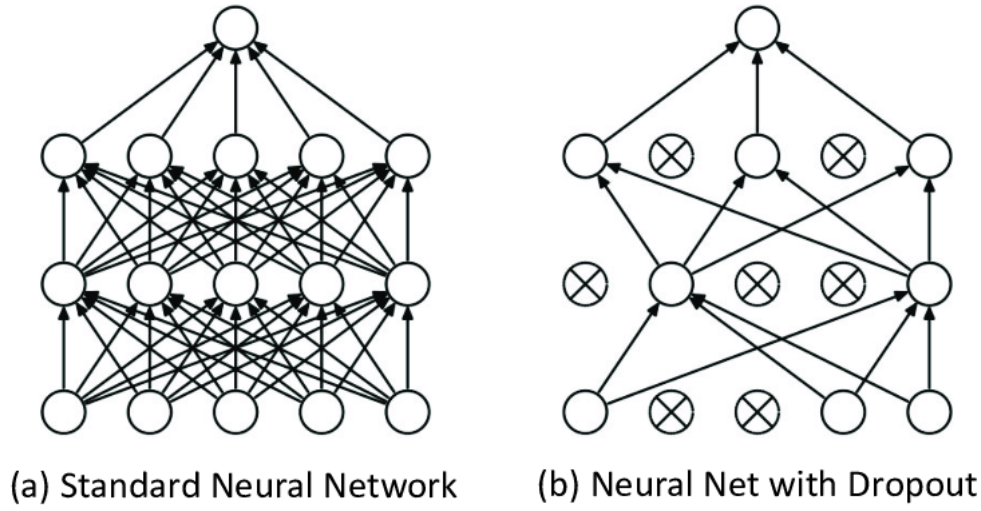


Gambar 2.7 *fully-connected layer*

Sumber: Afshine (2019)

2.7.2 *Dropout Regulation*

Dropout merupakan suatu cara atau teknik regulasi jaringan syaraf dengan tujuan memilih beberapa neuron secara acak dan tidak akn dipakai selama proses pelatihan atau bisa dikatakan bahwa neuron-neuron tersebut dihilangkan secara acak. Hal tersebut memiliki arti bahwa kontribusi neuron yang dihilangkan akan diberhentikan sementara jaringan dan bobot baru juga tidak diterapkan pada enuron pada saat melakukan *backpropagation*. Gambar 2.8 merupakan gambar dari proses *dropout.*



Gambar 2.8 *Dropout regulation*

Berdasarkan Gambar 2.8 pada bagian (a) merupakan jaringan syaraf biasa yang mempunyai dua lapisan tersembunyi. Sedangkan pada bagian (b) merupakan jaringan syaraf yang memakai *dropout*. Dari Gambar 2.8 dapat dilihat beberapa neuron aktivasi yang tidak digunakan lagi. Pemakaian teknik ini sangat mudah untuk diaplikasikan pada model CNN dan akan memiliki dampak pada perforam model dalam melatoh serta mengurangi *overfitting* (Srivastava, Hinton, G, & Kriszhevsky, A, 2014).

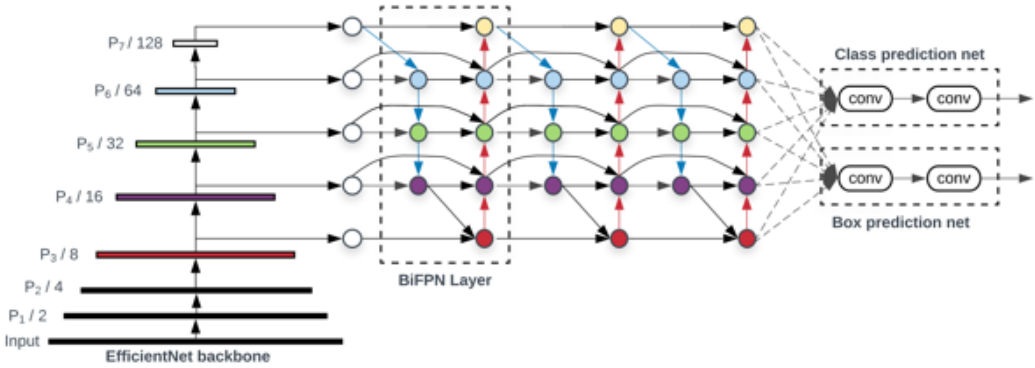
2.8 EfficientDet

EfficientDet merupakan salah satu model yang dapat digunakan untuk melatih sistem berbasis *deep neural network* dengan basis algoritma CNN. Model ini dapat melatih kumpulan data berupa gambar sehingga dapat digunakan untuk pengembangan deteksi objek. Dalam pengembangannya, model ini diciptakan untuk meningkatkan efisiensi pelatihan dengan

menggunakan parameter yang sedikit namun tetap dapat menghasilkan akurasi yang baik (Tan et al., 2020).

2.8.1 Arsitektur

Gambar 2.9 menunjukkan arsitektur model dari *efficientdet*. Sebagian besar dari model ini menggunakan metode deteksi jaringan satu tahap. Metode deteksi jaringan satu tahap akan melakukan prediksi pada seluruh wilayah gambar sehingga prosesnya dapat berjalan cepat. Kemudian, untuk jaringan *backbone* dari model ini menggunakan hasil pelatihan dari model *EfficientNet* yang dilatih dengan dataset *ImageNet.* Selanjutnya, jaringan fitur dalam model ini dinamakan dengan *bi-directional feature pyramid network* (BiFPN). Jaringan fitur tersebut mengambil fitur yang berada pada level 3-7 {P3, P4, P5, P6, P7}. Fitur tersebut berada pada jaringan *backbone* yang kemudian berulang kali diproses oleh BiFPN. Kemudian, fitur-fitur yang telah diproses diumpankan ke jaringan kelas dan kotak untuk menghasilkan prediksi kelas objek dan kotak pembatas masing-masing untuk objek.



Gambar 2.9 Arsitektur model *EfficientDet*

Sumber: Tan et al (2020)

2.8.2 *Compound Scaling*

Dalam *Efficientdet*, jaringan *backbone,* BiFPN, jaringan kotak prediksi, dan jaringan kelas prediksi diberlakukan penskalaan khusus terhadap lebar, kedalaman, dan resolusi citra yang bertujuan untuk menghasilkan kemampuan model deteksi yang baik pada tiap skala gambar serta memiliki efisiensi yang baik saat pelatihan (Tan et al., 2020).

Pada jaringan *backbone*, model ini tetap menggunakan lebar/kedalaman yang sama pada koefisien *Efficientnets* B0 – B7. Pada jaringan BiFPN penskalaan lebar dilakukan dengan

persamaan (2.1) dan penskalaan kedalaman dilakukan menggunakan persamaan (2.2). Selanjutnya, pada jaringan prediksi menggunakan penskalaan yang sama dengan jaringan BiFPN tetapi secara linier bertambah dengan perhitungan pada persamaan (2.3). Dan yang terakhir, pada masukan gambar menggunakan perhitungan pada persamaan (2.4). Nilai koefisien setiap model pada *efficientdet* dapat dilihat pada Tabel 2.1.

$$W_{bifpn} = 64\ x\ (1{,}35^{\emptyset}) \quad (2.1)$$

$$D_{bifpn} = 64 + \emptyset \quad (2.2)$$

$$D_{box} = D_{class} = 3 + [\frac{\emptyset}{3}] \quad (2.3)$$

$$R_{input} = 512 + \emptyset\ x\ 128 \quad (2.4)$$

Tabel 2.1 Arsitektur penskalaan *EfficientDet*

Model	*Input Layer*	*BiFPN Layers*	*ClassNet and BoxNet layers*
D0 ($\emptyset = 0$)	512	3	3
D1 ($\emptyset = 1$)	640	4	3
D2 ($\emptyset = 2$)	768	5	3
D3 ($\emptyset = 3$)	896	6	4
D4 ($\emptyset = 4$)	1024	7	4
D5 ($\emptyset = 5$)	1280	7	4
D6 ($\emptyset = 6$)	1280	8	5
D7 ($\emptyset = 7$)	1536	8	5

Selain itu, pada *efficientdet* fungsi aktivasi yang digunakan berupa *swish activation.* Fungsi aktivasi ini digunakan di setiap jaringan, yakni jaringan *backbone,* jaringan BiFPN, jaringan kelas prediksi dan jaringan kotak prediksi. Secara umum, fungsi aktivasi *swish* bekerja lebih baik pada arsitektur *deep neural network* (Tomas, 2021).

2.8.3 *EfficientDet-lite*

EfficientDet-lite merupakan model yang dioptimalkan untuk arsitektur perangkat bergerak. Model ini memiliki arsitektur dan pemrosesan yang sama dengan *efficientdet*. Di dalamnya terdapat lima model pralatih dari lite0 hingga lite 4. Tabel 2.2 menunjukkan detail

setiap model pralatih yang terdapat pada *efficientdet-lite.* Selain itu, perbedaan antara *efficientdet* dengan *efficientdet-lite* terletak pada fungsi aktivasi yang digunakan dan ukuran *input layers* yang digunakan. Ukuran *input layer* pada *efficientdet* dapat dilihat pada Tabel 2.1, sedangkan untuk *efficientdet-lite* dapat dilihat pada Tabel 2.2. Perbedaan selanjutnya, yakni pada *Efficientdet-lite* menggunakan fungsi aktivasi ReLu6 sedangkan pada *efficientdet* menggunakan fungsi aktivasi *swish*. ReLu6 menggunakan operasi matematika yang lebih sederhana dibandingkan dengan fungsi aktivasi *swish* sehingga ReLu6 dapat bekerja lebih efisien. Dengan menggunakan ukuran masukan yang lebih kecil dan fungsi aktivasi yang lebih sederhana, *efficientdet-lite* memiliki pemrosesan komputasi yang lebih cepat dan efisien sehingga cocok untuk diimplementasikan pada perangkat bergerak (Tomas, 2021).

Tabel 2.2 Arsitektur penskalaan *EfficientDet-lite*

Model	*Input Layer*	*BiFPN Layers*	*ClassNet and BoxNet layers*
lite0	320	3	3
lite1	384	4	3
lite2	448	5	3
lite3	512	6	4
lite4	640	7	4

2.9 Tensorflow

Tensorflow merupakan suatu *computational framework* yang dapat digunakan untuk membangun model *machine learning* seperti *neural network. Tensorflow* dikembangkan oleh *Google Brain Team* yang dirilis dengan lisensi *open-source Apache 2.0* pada 9 November 2015. *Tensorflow* merupakan suatu pustaka perangkat lunak yang terbuka untuk melakukan komputasi numerik dengan menggunakan grafik aliran data. Pada awalnya, pustaka ini dikembangkan untuk untuk keperluan menjalankan pembelajaran mesin dan riset *deep neural network. Tensorflow* kemudian dikembangkan dengan cara menggabungkan algoritma teknik pengoptimalan kompilasi dengan tujuan untuk mempermudah penghitungan ekspresi matematis yang banyak sehingga masalah terkait dengan waktu yang dibutuhkan untuk melakukan perhitungan dapat dikerjakan lebih cepat (Manajang et al., 2020).

2.10 Matrik Evaluasi

Performa dari model deteksi objek dapat ditentukan melalui parameter hasil pengukuran matrik evaluasi. Matrik evaluasi yang populer digunakan dalam deteksi objek adalah *Average Precision.* Untuk dapat memperoleh nilai AP diperlukan nilai perhitungan dari *precision* dan

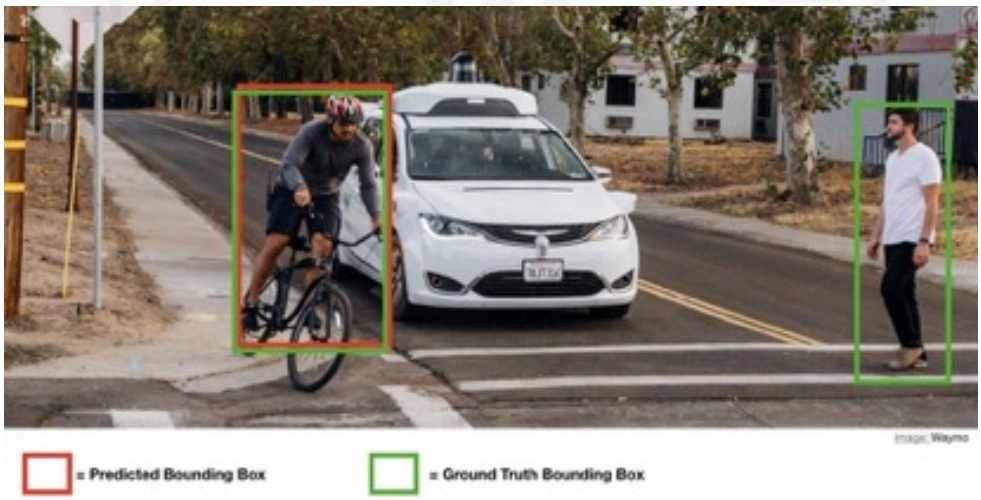
recall (Tan et al., 2020). Dengan kata lain, matriks tersebut membandingkan ketepatan prediksi dari keseluruhan data yang diterima dengan data yang sebenarnya. Perhitungan *precision* diperoleh dari persamaan (2.5) dan perhitungan *recall* diperoleh dari persamaan (2.6). Dari hasil perhitungan kedua nilai tersebut maka akan dilanjutkan untuk menghitung nilai AP *(Average Precision)* dengan cara menghitung rata-rata nilai *precision* dari keseluruhan *recall*. Perhitungan AP dapat dilihat pada persamaan (2.7).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2.5)$$

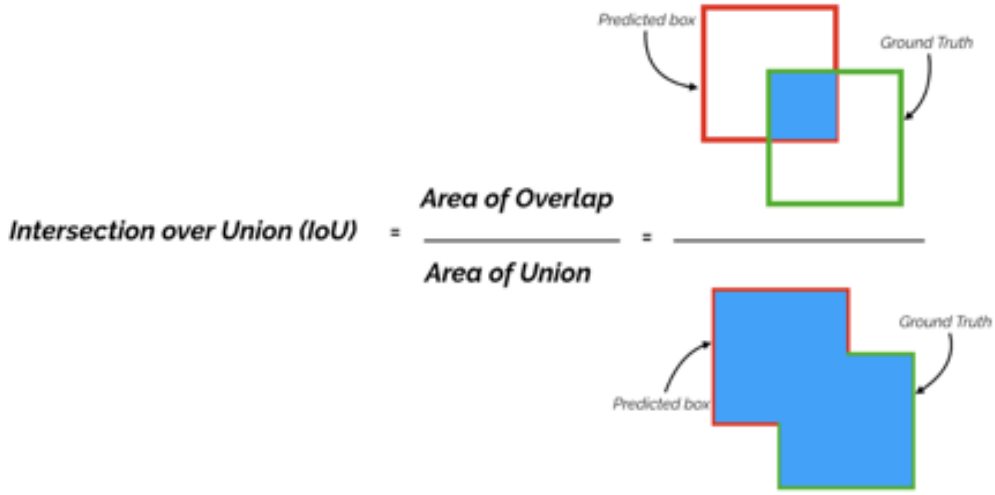
$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2.6)$$

$$AP = \frac{1}{11}\sum_{Recall_{i-11}} Precision(Recall_i) \quad (2.7)$$

Dalam deteksi objek, nilai dari *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN) menggunakan nilai ambang batas dari *Intersection of Union* (IoU). IoU merupakan nilai dari persimpangan antara kotak prediksi dengan kotak *Ground Truth.* Ilustrasi dari IoU dapat dilihat pada Gambar 2.10 dan perhitungannya dapat dilihat pada Gambar 2.11.

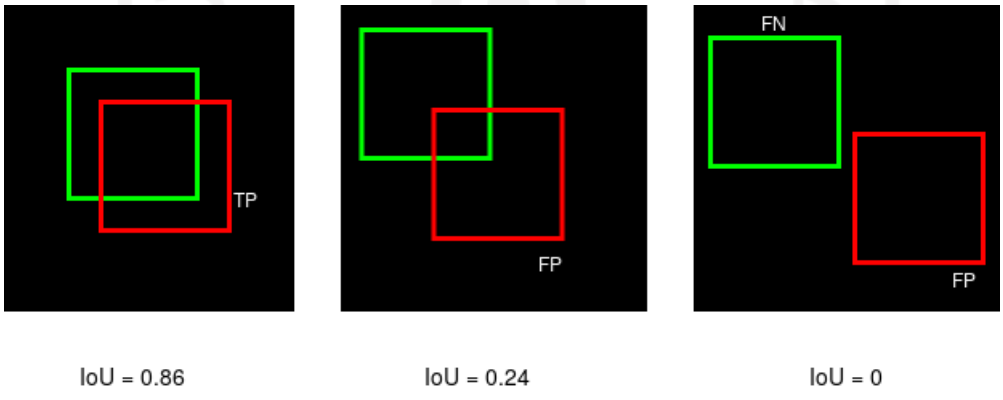


Gambar 2.10 Contoh kotak prediksi dan kota *ground truth*



Gambar 2.11 Ilustrasi perhitungan nilai IoU

Nilai sebuah deteksi dianggap sebagai *True Positive* (TP) hanya jika pendeteksiannya memang benar terdapat objek dengan klasifikasi yang tepat dan nilai IoU lebih besar dari ambang batas yang telah ditentukan, misalnya IoU lebih besar dari atau sama dengan 0.5. *False Positive* (FP) berarti hasil deteksi objek salah, tidak sesuai dengan klasifikasi, dan nilai IoU kurang dari nilai ambang batas. *False Negative* (FN) saat hasil deteksi objek tidak terdeteksi sama sekali. Agar bisa lebih dipahami berikut ilustrasi penentuan setiap nilai dengan IoU sebesar 0.5 pada Gambar 2.12



Gambar 2.12 Penentuan TP, FP, dan FN

Keterangan:

True Positive (TP) = Jumlah data kelas positif diprediksi benar sebagai kelas positif

False Negative (FN) = Jumlah data kelas positif diprediksi salah sebagai kelas negative

True Negative (TN) = Jumlah data kelas negatif diprediksi benar sebagai kelas negatif
False Positive (FP) = Jumlah data kelas negatif diprediksi salah sebagai kelas positif

2.11 Bias Dataset

Bias data dalam deteksi objek merupakan data yang tidak merepresentasikan suatu objek secara benar. Dunia visual sangat kompleks dan begitu bervariasi sehingga kumpulan sampel yang tak terbatas hanya akan merepresentasikan beberapa aspek saja. Selain itu, jika data yang dikumpulkan untuk kebutuhan tertentu, kumpulan gambar hanya akan mencakup beberapa wilayah visual tertentu. Oleh karena itu, tidak mengherankan apabila kumpulan gambar yang sudah ditentukan sebelumnya *(pre-defined),* seperti kumpulan data visi komputer yang ada, menghadirkan bias tertentu di dalam dataset (Tommasi et al., 2017). Terdapat dua jenis bias yang umumnya terjadi dalam deteksi objek, diantaranya sebagai berikut:

a. Bias penangkapan/pengambilan, bias ini berhubungan dengan bagaimana gambar diperoleh baik dari segi perangkat yang digunakan maupun sudut pengambilan gambar, kondisi pencahayaan, dll.
b. Bias pelabelan, pada bias ini, kategori atau label bias didapatkan dari kesalahan pemberian nama objek yang tepat. Gambar serupa mungkin dianotasikan dengan nama yang berbeda, sementara karena variabilitas dalam kelas, nama yang sama bisa menunjukkan gambar berbeda secara visual. Akhirnya, setiap kelas mungkin berisi satu set kategori yang berbeda dan ini menyebabkan terjadinya bias negatif. Secara keseluruhan, masalah bias kumpulan data *(dataset bias)* berasal dari penyebab yang telah disebutkan. Keterbatasan dari kumpulan data yang ada dibuat untuk mengevaluasi model deteksi objek.

2.12 Studi Literatur

Melakukan studi literatur terhadap penelitian sebelumnya yang memiliki keterkaitan topik perlu dilakukan agar pemahaman terkait topik yang sedang diteliti semakin bagus dan terhindar dari penjiplakan atau plagiasi dari penelitian sebelumnya. Tinjauan pustaka yang telah dilakukan dalam penelitian ini berkaitan dengan *Object Detection*, pengumpulan dataset, arsitektur model, dan bias dataset. Literatur penelitian dilakukan pada 7 makalah penelitian tentang perancangan dataset untuk keperluan deteksi objek dan 4 makalah penelitian tentang bias yang ada dalam dataset dan satu makalah terkait dengan arsitektur model. Makalah yang didapatkan menggunakan berbagai macam algoritma yang berbasis CNN. Tabel 2.3 memaparkan teknik pengumpulan data dan anotasi yang telah dilakukan oleh peneliti

sebelumnya dan pada Tabel 2.4 memaparkan bias yang ada dalam dataset dan cara menangani bias tersebut.

Penelitian tentang pengembangan arsitektur model deteksi objek telah banyak dilakukan oleh peneliti sebelumnya. Tan et al melakukan pembuatan model pra-latih deteksi objek dengan sumber *backbone* yang berasal dari hasil pelatihan model *Efficientnet* menggunakan dataset *Imagenet*. Model ini memberikan solusi atas kebutuhan efisiensi yang penting dalam dunia visi komputer. Dalam penelitian tersebut, *Feature Pyramid Networks* (FPNs) yang digunakan menggunakan jaringan fitur dengan metode baru, yakni *bi-directional feature pyramid network* (BiFPN) yang menggabungkan fitur multiskala dengan cepat dan mudah dan mengusung metode penskalaan di setiap jaringan, mulai dari jaringan ekstraksi fitur hingga jaringan pendeteksian secara bersamaan. Model *EfficientDet-D7* memperoleh nilai AP sebesar 55,1 AP dengan menggunakan matriks evaluasi COCO.

Pembuatan dataset yang bersumber dari internet seperti *Flicker* dan *PapperDoll* telah dilakukan oleh Shao et al dan Zheng et al. Dalam kedua dataset tersebut terdapat perbedaan cara pemilihan gambar dan anotasi data. Dataset yang dibuat oleh Shao et al memilih gambar dengan kriteria *iconic images* yang artinya gambar yang terpilih tidak berfokus pada satu objek serta teknik anotasi yang diterapkan menggunakan *bounding box* dengan menerapkan aturan klasifikasi dan aturan *bounding box.* Aturan klasifikasi memiliki keterkaitan dengan pemberian label untuk menghindari ambiguitas dan autran *bounding box* untuk memberikan kesepakatan area yang akan dianotasi pada setiap kelas (Shao et al., 2019). Kemudian pada dataset yang dibuat oleh Zheng et al hanya memilah gambar berdasarkan kualitas kemudian menggunakan teknik anotasi poligon (Zheng et al., 2018). Hasil yang diperoleh dari kedua pembuatan dataset ini menunjukkan bahwa dataset buatan Shao et al menghasilkan nilai mAP yang lebih baik, yakni 90.

Pembuatan dataset yang dilakukan oleh Xia et al, Tourani et al, de Tournemire et al, Spagnolo et al, dan Maeda et al secara umum menggunakan teknik pengumpulan data yang sama dengan cara pengambilan secara langsung dengan menggunakan kamera atau suatu sensor. Saat pengambilan gambar, sudutnya harus disesuaikan dengan sudut pengambilan gambar saat uji coba atau praktek di lapangan (Maeda et al., 2018). Pada dataset Xia et al, variasi dalam dataset ditingkatkan dengan cara mengumpulkan data dari berbagai kota dalam waktu yang berbeda-beda. Kemudian, metode anotasi yang digunakan berupa (x_c, y_c, w, h, θ), dimana θ merupakan sudut dari *bounding box*. Dengan menerapkan teknik metode anotasi tersebut, anotator dapat melakukan anotasi yang baik terhadap objek yang memiliki orientasi

miring dan mengurangi *overlap* yang berlebih dengan anotasi yang biasa. Nilai mAP yang diperoleh dengan anotasi tersebut sebesar 54.13 (Xia et al., n.d.). Kemudian, pada pembuatan dataset yang dilakukan oleh Tourani et al menggunakan teknik anotasi yang dilakukan secara otomatis dengan bantuan alat yang bernama *Ybat*. Nilai yang diperoleh sebesar 87.4804 (Tourani et al., 2021).

Tabel 2.3 Perbandingan literatur tentang dataset deteksi objek

Penulis	Judul	Teknik Pengumpulan Data	Teknik Anotasi
(Shao et al., 2019)	*Objects365: A large-scale, high-quality dataset for object detection*	• Data didapatkan dari *Flicker.* • Memilah gambar yang termasuk ke dalam kriteria *iconic images.*	Memberikan kategori, dan melakukan anotasi sesuai dengan kelasnya kemudian anotasi dilakukan dengan menerapkan aturan klasifikasi dan *bounding box*
(Zheng et al., 2018)	*ModaNet: A large-scale street fashion dataset with polygon annotations*	• Data didapatkan dari dataset *PaperDoll.* • Memilah 2000 gambar yang memiliki kualitas rendah dan 2000 gambar dengan kualitas tinggi.	Gambar pada dataset dianotasi dengan teknik poligon. Kemudian, anotator dapat melewati gambar yang menurutnya ambigu untuk tidak dianotasi.
(Xia et al., n.d.)	*DOTA: A Large-Scale Dataset for Object Detection in Aerial Images | Enhanced Reader*	• Gambar yang dikumpulkan berupa gambar yang diambil dari udara. • Penangkapan gambar menggunakan beragam sensor dan bersumber dari *platform* (contohnya *Google Earth).* • Variasi data ditingkatkan dengan cara mengumpulkan data dari berbagai kota dalam waktu yang berbeda-beda.	Metode anotasi yang digunakan berupa (x_c, y_c, w, h, θ), dimana θ merupakan sudut dari *bounding box*.
(Tourani et al., 2021)	*Iranis: A Large-scale Dataset of Farsi License Plate Characters*	• Pengambilan secara langsung menggunakan kamera yang tidak dikalibrasi. • Variasi gambar didapatkan dari perbedaan kondisi cahaya, perbedaan resolusi, kontras, jarak, dan sudut pemotretan.	Proses anotasi data dilakukan secara otomatis dengan validasi bertahap. Aplikasi yang digunakan adalah *Ybat*.
(de Tournemire et al., 2020)	*A Large Scale Event-based Detection Dataset for Automotive*	• Mengumpulkan gambar dari potongan video yang direkam selama perjalanan di mobil. • Pengambilan gambar menggunakan kamera yang disebut dengan *event camera.* • Kamera memiliki hasil pengukuran *gray-level.*	Anotasi dilakukan dengan cara pemberian *bounding box* terhadap setiap objek yang ada pada gambar dan memberikan label sesuai dengan kelasnya.

Penulis	Judul	Teknik Pengumpulan Data	Teknik Anotasi
		• Variasi gambar diperoleh dari perbedaan jalan yang dilalui mobil.	
(Spagnolo et al., 2019)	*A new annotated dataset for boat detection and re-identification*	• Pengumpulan gambar dari kamera yang diletakkan pada sebuah tower. • Pemotongan manual untuk mendapat objek kapal. • Menggunakan Mask-RCNN dan *library* ImgAug dalam proses pengolahan gambar.	
(Maeda et al., 2018)	*Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone*	• Pengambilan gambar menggunakan *smartphone* yang diletakkan pada *dashboard* mobil. • Dataset digunakan untuk keperluan mendeteksi jalan yang rusak. • Sudut pengambilan gambar dilakukan dengan cara menyesuaikan sudut saat proses praktik deteksi objek.	

Tabel 2.4 Perbandingan literatur tentang bias data pada deteksi objek

Penulis	Judul	Identifikasi Bias pada Dataset	Teknik Mengatasi Bias pada Dataset
(Kortylewski, Egger, Schneider, et al., 2018), ,	*Empirically analyzing the effect of dataset biases on deep face recognition systems*	Bias yang terjadi pada dataset deteksi wajah berupa tidak lengkapnya perubahan sudut wajah dan arah datangnya cahaya.	Memanfaatkan alat bantu berupa *Face Image Generator* untuk membuat data sintesis wajah.
(Kortylewski, Egger, Morel-Forster, et al., 2018)	*Can Synthetic Faces Undo the Damage of Dataset Bias to Face Recognition and Facial Landmark Detection?*		
(Derman, n.d.)	*Dataset Bias Mitigation Through Analysis of CNN Training Scores*	Adanya satu jenis gambar yang dominan pada suatu kelas.	Membuat kelas baru yang spesifik. Contohnya, wajah pria dibuat menjadi dua kelas untuk pria berkulit terang dan berkulit gelap.
(Tommasi et al., 2017)	*A deeper look at dataset bias*	Bias yang ditemukan berupa bias penangkapan dan label bias.	Menerapkan fitur DeCAF dan BOWsift dalam mempelajari bias yang ada dalam dataset.

Dari hasil kajian literatur terkait pembuatan dataset pada Tabel 2.3, diperoleh sebanyak tujuh dataset berbeda yang digunakan dalam pelatihan deteksi objek, yakni diantaranya digunakan untuk keperluan mendeteksi baju, mendeteksi jalan berlubang, mendeteksi area bumi, mendeteksi objek secara umum, mendeteksi kapal, mendeteksi jalanan, dan mendeteksi mendeteksi plat nomor kendaraan. Penelitian sebelumnya telah melakukan pembuatan dataset

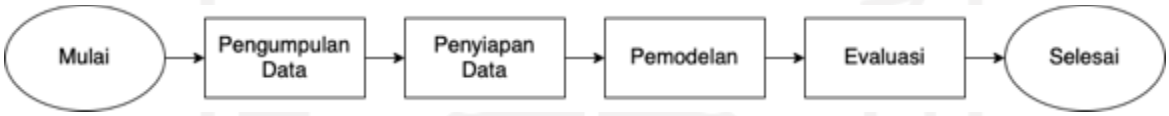
untuk berbagai macam keperluan yang berbeda. Dari hasil literatur tersebut, terdapat beberapa teknik dalam pengumpulan data, yakni dengan cara mengambil data yang tersedia di internet, mengambil data secara langsung dengan kamera, atau dari hasil rekaman video. Kemudian, untuk teknik anotasi diperoleh sebanyak empat cara yang dapat dilakukan, diantaranya adalah melakukan anotasi secara otomatis menggunakan aplikasi, pemberian *bounding box*, *bounding box* dengan sudut, dan anotasi poligon.

Kemudian, hasil kajian literatur dari Tabel 2.4 diperoleh hasil berupa tiga faktor yang dapat menyebabkan bias pada dataset. Faktor tersebu diantranya, tidak lengkapnya sudut pengambilan gambar, adanya satu jenis gambar yang dominan, dan bias pelabelan. Kemudian terdapat empat cara untuk mengatasi bias yang ada pada dataset. Cara yang dapat dilakukan adalah menggunakan aplikasi, membuat kelas baru, menerapkan fitur DeCaf dan BOWsift. Menurut hasil kajian dari makalah Kortylewski, Egger, Schneider, et al, Tommasi et al, dan Derman, n.d., bias yang ada dalam dataset akan selalu ada.

Berdasarkan studi literatur yang terdapat pada Tabel 2.3, Tabel 2.4 dan uraian singkat yang telah dilakukan. Penelitian ini akan menggunakan beberapa teknik pengumpulan data dengan cara pengambilan data di toko *retail* menggunakan kamera *smartphone*, kemudian melakukan anotasi dengan cara pemberian *bounding box*, menerapkan bias arah datangnya cahaya, bias pengambilan gambar, dan bias pelabelan. Perbedaan pada penelitian ini dengan penelitian yang terdahulu adalah terletak pada penggunaan model. Pada penelitian ini akan menggunakan model arsitektur *efficientdet-lite* yang telah dioptimalkan untuk perangkat bergerak.

BAB III
METODOLOGI PENELITIAN

Pada bab ini, peneliti akan menerangkan metodologi penelitian yang diterapkan untuk menyelesaikan rumusan masalah yang akan diteliti. Metodologi pada penelitian ini berisi tahapan-tahapan yang akan dilalui peneliti untuk mencapai hasil yang diharapkan. Kemudian, terdapat empat tahapan yang harus dilalui, yakni perancangan dataset, perancangan sistem, pelatihan dataset, dan pengujian. Tahapan tersebut dapat dilihat pada Gambar 3.1



Gambar 3.1 Bagan alir metode penelitian

3.1 Pengumpulan Data

Dataset merupakan data yang diperlukan dalam proses pelatihan model. Pada model *efficientdet-lite,* dataset yang diperlukan berupa data gambar. Dalam penelitian ini, dataset dipersiapkan untuk bisa mendeteksi empat belas macam produk *retail* yang berbeda. Produk yang akan dideteksi pada penelitian ini dapat dilihat pada Gambar 3.2. Dataset yang dirancang akan menerapkan beberapa jenis bias data, yakni bias arah datangnya cahaya, bias sudut pengambilan gambar, dan bias pelabelan. Penerapan bias tersebut bertujuan untuk mengetahui pengaruh setiap bias data terhadap performa akurasi deteksi objek dengan menggunakan model *efficientdet-lite* yang akan diimplementasikan pada perangkat bergerak.

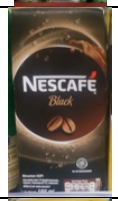
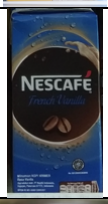
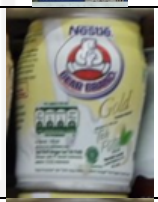
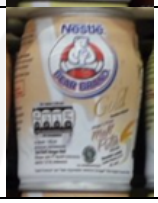


Gambar 3.2 Data produk *retail*

3.1.1 Populasi dan Sampel

Populasi dalam penelitian ini menggunakan produk *retail* yang ada di toko. Kemudian, sampel yang digunakan dalam penelitian ini menggunakan empat belas jenis produk yang berada dalam kategori *ready to drink* (RTD). Kategori tersebut merupakan produk minuman yang dapat langsung diminum oleh konsumen. Tabel 3.1 menunjukkan objek serta nama kelas yang digunakan dalam penelitian ini.

Tabel 3.1 Nama kelas dan penampakan objek

No.	Nama Kelas	Objek	No.	Nama Kelas	Objek
1.	NESCAFE Latte 24x220ml ID		8.	Milo Original	
2.	NESCAFE Cappuccino 24x220ml ID		9.	Milo Calcium	
3.	NESCAFE Caramel Macchiato 24x220ml ID		10.	NS UHT Black	
4.	Milo NutriUp		11.	NS UHT CoffeCream	
5.	NESCAFE Original		12.	NS UHT FrenchVanilla	
6.	NESCAFE Latte Can		13.	White Tea	
7.	NESCAFE Mocha Can		14.	White Malt	

3.1.2 Jenis dan Sumber Data

Dalam penelitian ini, peneliti menggunakan jenis data primer. Data tersebut diperoleh dengan cara mengambil foto produk secara langsung di beberapa toko menggunakan kamera utama *smartphone*. Foto yang diambil harus memenuhi beberapa kriteria yang digunakan

dalam penelitian ini. Gambar 3.3 merupakan hasil pengambilan gambar sesuai dengan kriteria. Kriteria pengambilan foto tersebut diantaranya:

a. Objek produk harus tampak jelas di dalam gambar dan tidak boleh ada benda lain yang menghalangi objek.
b. Pengambilan foto berjarak 25 cm hingga 100 cm dari objek.
c. Sudut pengambilan foto mulai dari sudut atas, bawah, kiri, kanan, serta sejajar dengan objek.
d. Mengambil foto objek yang memantulkan cahaya.



Gambar 3.3 Hasil pengambilan gambar

3.2 Penyiapan Data

Pada tahapan ini, peneliti akan mempersiapkan dan memproses data yang sudah dikumpulkan. Proses-proses yang akan dilalui adalah praproses gambar, pembentukan kondisi dataset, dan pemberian label. Pada praproses gambar, dilakukan penurunan resolusi gambar.

Selanjutnya, proses pembentukan kondisi dataset akan menghasilkan empat jenis dataset yang berbeda, yakni dataset Optimal, Bias1, Bias2, dan Bias3. Kemudian, proses pemberian label dilakukan terhadap objek yang ada pada setiap gambar. Tabel 3.2 menunjukkan jumlah gambar yang diproses pada setiap jenis dataset.

Tabel 3.2 Jumlah gambar pada tiap jenis dataset

Jenis Dataset	Jumlah Gambar
Optimal	605
Bias1	545
Bias2	605
Bias3	600

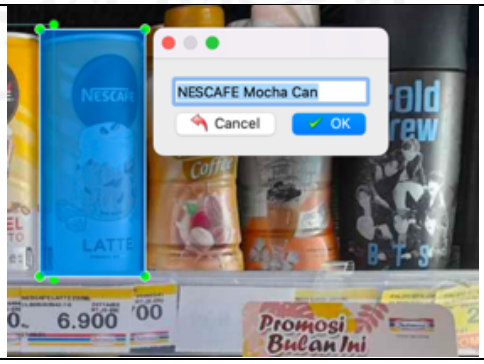
3.2.1 Praproses Gambar

Pada tahap ini, citra yang sudah terkumpul akan diolah agar tidak terjadi penggunaan memori yang berlebih saat proses pelatihan dataset. Citra yang terkumpul akan disamakan resolusinya dengan ukuran terpanjang 1024-pixel kemudian mempertahankan aspek rasio dari semua citra. Penurunan resolusi tersebut bisa menghemat penggunaan memori untuk penyimpanan citra serta menghemat penggunaan memori saat pelatihan data.

3.2.2 Pembentukan Kondisi Dataset

Pada tahap ini dilakukan pembuatan dataset dengan menerapkan bias untuk mengetahui pengaruhnya terhadap performa deteksi objek. Jenis bias yang digunakan berupa, bias arah datangnya cahaya, bias penangkapan, dan bias label sehingga akan dihasilkan tiga model dengan dataset yang memiliki bias dan satu model dengan dataset yang tidak memiliki bias. Peneliti akan membandingkan setiap pengaruh kondisi bias terhadap performa yang dihasilkan oleh model. Untuk lebih mudah diingat, Tabel 3.3 menunjukkan nama ke empat dataset, jenis bias yang diterapkan beserta dengan contoh gambarnya pada dataset.

Tabel 3.3 Jenis dataset dan jenis bias yang diterapkan

Nama Dataset	Jenis Bias	Contoh Gambar
Optimal	Tidak ada	
Bias1	Bias arah datangnya cahaya	
Bias2	Bias sudut penangkapan	
Bias3	Bias label	

Dataset Optimal

Pada kondisi dataset yang pertama, peneliti akan membuat dataset dengan cara mengambil gambar secara berhadapan langsung dengan produk tersebut. Pengambilan gambar diambil dengan tiga jarak yang berbeda, yakni satu langkah sampai dengan tiga langkah dari produk tersebut. Dalam dataset ini, bias terhadap citra seperti, bias arah datangnya cahaya dan bias pengambilan gambar akan diminimalisir untuk memperoleh nilai akurasi yang optimal. Selain itu, hasil nilai AP dari model yang menggunakan dataset ini akan dijadikan sebagai acuan untuk mengukur pengaruh bias terhadap dataset. Dataset yang pertama dinamakan dengan dataset Optimal. Ciri-ciri dari dataset ini, yakni pengambilan gambar tegak lurus dengan objek, pencahayaan yang cukup agar tidak terjadi pantulan cahaya dan kelebihan cahaya (*over exposure*), serta tidak ada kesalahan label. Contoh data gambar pada dataset Optimal dapat dilihat pada Gambar 3.4.



Gambar 3.4 Data gambar optimal

Dataset Bias1

Pada kondisi dataset yang kedua, jenis bias yang diterapkan pada dataset ini berupa bias arah datangnya cahaya. Pada dataset ini, akan dikumpulkan citra dengan kondisi cahaya yang mengarah langsung ke produk sehingga menimbulkan pantulan cahaya pada produk. Pantulan cahaya tersebut mengakibatkan objek menjadi tidak representatif sehingga menimbulkan bias. Jarak pengambilan gambar pada citra ini sama dengan kondisi dataset Optimal. Jumlah citra yang akan dikumpulkan pada dataset ini sebanyak 50% dari jumlah citra pada dataset. Contoh bias pada dataset ini dapat dilihat pada Gambar 3.5. Pada kasus ini tidak semua objek memantulkan cahaya sehingga peneliti mencoba untuk menurunkan tingkat kecerahan citra yang bisa dilihat pada Gambar 3.6.



Gambar 3.5 Kondisi bias pertama



Gambar 3.6 Penurunan kecerahan citra

Dataset Bias2

Pada dataset Bias2, jenis bias yang ada dalam dataset ini berupa tidak sesuainya teknik pengambilan gambar pada saat pengumpulan dataset dengan teknik pengambilan gambar saat pengujian dataset. Oleh karena itu, peneliti akan mengambil foto dengan berbagai macam sudut pengambilan gambar dengan menerapkan jarak yang sama pada dataset pertama. Jumlah citra yang akan dikumpulkan menggunakan teknik ini sebanyak 50% dari jumlah citra yang ada pada dataset. Contoh bias pada dataset ini dapat dilihat pada Gambar 3.7.



Gambar 3.7 Kondisi bias kedua

Dataset Bias3

Pada dataset yang terakhir atau yang keempat, dataset ini dinamakan dengan dataset Bias3. Jenis bias yang digunakan berupa bias pelabelan, satu objek dapat diberikan banyak label dan satu label dapat dipakai pada beragam objek. Perlakuan yang akan diterapkan pada dataset ini berupa memberikan pelabelan yang salah terhadap setiap objek pada 50% citra pada setiap kelas. Pada kondisi dataset ini, peneliti menggunakan kondisi yang sama dengan dataset pertama. Contoh biasnya dapat dilihat pada Gambar 3.8, setiap objek dapat diberikan label yang sama.



Gambar 3.8 Kondisi bias ketiga

3.2.3 Pelabelan Objek

Proses pelabelan objek yang terdapat pada citra bertujuan untuk memberikan keterangan terhadap objek yang ditargetkan sebelum melakukan proses pelatihan data. Selanjutnya, saat proses pelatihan dilakukan, model mempelajari setiap objek yang telah diberikan label. Pada penelitian ini, pelabelan dilakukan dengan perangkat lunak yang bernama *LabelImg* yang dapat dilihat pada Gambar 3.9. Proses pelabelan dilakukan dengan cara memberikan sebuah *bounding box* dan nama kelas atau label terhadap objek yang akan diberikan label. *Bounding box* merupakan sebuah segi empat yang melingkupi suatu objek pada gambar yang berfungsi sebagai penanda suatu objek secara spesifik di dalam gambar.



Gambar 3.9 Pemberian label dengan *LabelImg*

Kemudian, *LabelImg* akan menyimpan koordinat setiap sudut *bounding box* dan label objek ke dalam file xml. Isi file xml tersebut dapat dilihat pada Gambar 3.10. Baris ke-9 menunjukkan sisi terpanjang dari gambar yang dianotasi. Baris ke-14 sampai ke-25 merupakan baris yang berisi tentang informasi *bounding box*, diantaranya baris ke-15 merupakan nama label yang digunakan dan baris ke-20 sampai ke-23 merupakan informasi koordinat dari setiap sudut *bounding box*.

```
1. <annotation>
2.  <folder>05 nescafe baru 3</folder>
3.  <filename>20210618_090459_compress36 - Putri Widyas.jpg</filename>
4.  <path>/Users/ridhotiyar/Downloads/object-detection-master/masak/05
   nescafe baru 3/20210618_090459_compress36 - Putri Widyas.jpg</path>
5.  <source>
6.     <database>Unknown</database>
7.  </source>
8.  <size>
9.     <width>1280</width>
10.       <height>960</height>
11.       <depth>3</depth>
12.    </size>
13.    <segmented>0</segmented>
14.    <object>
15.       <name>NESCAFE Caramel Macchiato 24x220ml ID</name>
16.       <pose>Unspecified</pose>
17.       <truncated>0</truncated>
18.       <difficult>0</difficult>
19.       <bndbox>
20.          <xmin>134</xmin>
21.          <ymin>58</ymin>
22.          <xmax>427</xmax>
23.          <ymax>679</ymax>
24.       </bndbox>
25.    </object>
26.    <object>
27.       <name>NESCAFE Caramel Macchiato 24x220ml ID</name>
28.       <pose>Unspecified</pose>
29.       <truncated>0</truncated>
30.       <difficult>0</difficult>
31.       <bndbox>
32.          <xmin>431</xmin>
33.          <ymin>70</ymin>
34.          <xmax>670</xmax>
35.          <ymax>686</ymax>
36.       </bndbox>
37.    </object>
38. </annotation>
```

Gambar 3.10 Isi file xml

Dari proses pemberian label ini, didapatkan jumlah label dari setiap kelas dan setiap jenis dataset yang dapat dilihat pada Tabel 3.4.

Tabel 3.4 Jumlah label setiap kelas dan setiap dataset

No.	Nama Kelas	Dataset Optimal	Dataset Bias1	Dataset Bias2	Dataset Bias3
1.	NESCAFE Latte 24x220ml ID	134	116	125	138
2.	NESCAFE Cappuccino 24x220ml ID	110	112	186	130
3.	NESCAFE Caramel Macchiato 24x220ml ID	123	162	188	146
4.	Milo NutriUp	253	216	230	275
5.	NESCAFE Original	180	209	185	139
6.	NESCAFE Latte Can	198	192	183	140
7.	NESCAFE Mocha Can	185	192	149	131
8.	Milo Original	186	202	246	234
9.	Milo Calcium	235	192	199	209
10.	NS UHT Black	270	229	189	246
11.	NS UHT CoffeCream	167	136	114	188
12.	NS UHT FrenchVanilla	123	103	113	137
13	White Tea	243	220	162	270
14.	White Malt	146	217	229	247
Total label		2543	2511	2498	2630

3.3 Penyiapan Sistem Penelitian

Sistem deteksi objek pada penelitian ini akan menggunakan server *high computing UII* yang diberi nama hanabi. Sistem tersebut bertugas untuk menjalankan model *EfficientDet-Lite.* Objek yang akan diteliti berupa empat belas macam produk *retail*. Agar dapat terhubung ke server, peneliti menggunakan laptop pribadi untuk dapat memberikan perintah ke server. Kemudian, perangkat lunak yang diperlukan dalam mendukung penelitian ini berupa *Python3.8.8, Anaconda, JupyterHub, Cuda-11 Nvidia, dan cudnn 8.5.0.* Bahasa pemrograman yang digunakan adalah *python* dengan editornya *JupyterNotebook. Cuda dan cudnn* berfungsi sebagai perangkat lunak yang dapat memanfaatkan kemampuan *Graphics Processing Unit* (GPU) agar proses komputasi menjadi lebih cepat. Selain itu, penelitian ini memanfaatkan pustaka *Tensorflow* yang disediakan oleh google. Pustaka tersebut memiliki banyak fungsi yang bisa dimanfaatkan dalam penelitian terkait *machine learning, deep learning,* maupun *artificial intelligence.*

3.4 Pemodelan

Pada tahap ini, peneliti akan melakukan pemodelan sebanyak empat kali dengan menggunakan jenis dataset yang sudah disiapkan pada tahap penyiapan data. Kemudian, masing-masing dataset tersebut akan dilanjutkan untuk masuk ke tahap pelatihan hingga dihasilkan model yang dapat berjalan di perangkat bergerak.

3.4.1 *Split Data*set

Pada tahap ini akan dilakukan pembagian dataset menjadi *train set* dan *validation set*. *Train set* merupakan kumpulan data yang akan digunakan untuk *training model.* Sedangkan, *validation set* merupakan data yang digunakan untuk validasi model dan mencegah *overfitting. Overfitting* merupakan keadaan saat model terlalu fokus pada kumpulan data tertentu sehingga tidak dapat melakukan prediksi dengan tepat ketika diberikan data yang tidak serupa. Dalam pembagiannya, penelitian ini menggunakan rasio pembagian 20% untuk *validation set* dan 80% untuk *train set*.

3.4.2 *Training*

Sebelum pelatihan dimulai, peneliti perlu memanggil beberapa pustaka, yakni pustaka numpy dan tflite model maker. Pustaka numpy diperlukan untuk melakukan perhitungan saat evaluasi model dan pustaka tflite model maker digunakan untuk memanggil model, ekspor model, dan memulai pelatihan. Pemanggilan pustaka tersebut dapat menggunakan baris ke-1 hingga ke-5 pada Gambar 3.11. Kemudian, kode baris ke-7 berfungsi untuk melakukan pemanggilan model. Selanjutnya, kode pada baris ke-8 berfungsi untuk memasukkan nama label yang akan digunakan. Daftar nama label yang akan digunakan dapat dilihat pada Tabel 3.1. Berikutnya, peneliti perlu menentukan lokasi dari *train set* dan *validation set*. Baris kode ke-9 berfungsi untuk menentukan lokasi penyimpanan *train set* dan baris kode ke-10 berfungsi untuk menentukan lokasi penyimpanan *validation set*. Pada baris kode ke-9 dan ke-10, ada 3 parameter yang perlu diisi, yakni lokasi penyimpanan gambar, lokasi penyimpanan file xml, dan daftar label yang telah ditentukan pada baris ke-8. Setelah semuanya diisi, maka sistem dapat memulai pelatihan dengan memanggil baris kode yang terdapat pada Gambar 3.12.

```
1. import numpy as np
2.
3. from tflite_model_maker.config import ExportFormat
4. from tflite_model_maker import model_spec
5. from tflite_model_maker import object_detector
6.
7. spec = model_spec.get('nama model')
8. label = ['nama label']
9. train_data = object_detector.DataLoader.from_pascal_voc('lokasi
   gambar data train','lokasi xml data train',label_map=label)
10. validation_data = object_detector.DataLoader.from_pascal_voc('lokasi
   gambar validasi','lokasi xml validasi',label_map=label)
```

Gambar 3.11 Baris kode persiapan *training*

Arsitektur

Pada penelitian ini, akan digunakan arsitektur *EfficientDet-Lite* yang memiliki model lite0 s.d. lite4. Model yang akan digunakan adalah model lite2 berdasarkan penelitian yang telah dilakukan oleh Tomas Repak. Pada penelitian tersebut, Tomas melakukan pengujian menggunakan model lite0, lite2, dan lite3 yang diujikan dengan 20288 gambar (Tomas, 2021). Hasil pengujian tersebut dapat dilihat pada Tabel 3.5. Dari hasil penelitian tersebut, peneliti memilih menggunakan *efficientdet-lite2* karena memiliki nilai mAP yang tidak berbeda jauh dengan *efficientdet-lite3* tetapi memiliki selisih waktu respon yang cukup besar. Sehingga peneliti memutuskan untuk menggunakan model *efficientdet-lite2.*

Tabel 3.5 Hasil pengujian Tomas Repak

Model	Waktu respon rata-rata (ms)	mAP
Efficientdet-lite0	398,3	0,266
Efficientdet-lite2	985,2	0,346
Efficientdet-lite3	1581,4	0,380

Selanjutnya, Gambar 3.12 merupakan baris kode yang digunakan untuk memulai pelatihan dataset serta *hyperparameter* yang digunakan dalam pelatihan. Di dalam baris kode "*object_detector.create( )"* terdapat beberapa parameter yang dibutuhkan dalam proses pelatihan model *EfficientDet-Lite*. Baris kedua menunjukkan model yang akan digunakan dalam pelatihan dataset, peneliti akan menggunakan model *efficientdet-lite2*. Pada baris ketiga menunjukkan jumlah *batch size* yang akan digunakan, untuk ukuran *batch size*, peneliti menggunakan ukuran sebesar 30. *Batch size* merupakan banyaknya sampel yang disebarkan ke dalam arsitektur *neural network*. Kemudian, pada baris keempat menunjukkan jumlah *epochs* yang akan digunakan, jumlah *epochs* sebesar 145. *Epochs* merupakan perulangan yang dilakukan untuk meninjau dataset dengan parameter yang telah ditentukan. Treakhir, pada baris

kelima terdapat parameter *"train_whole_model",* peneliti menggunakan *Train whole model* yang disediakan oleh *efficentdet-lite* agar pelatihan yang dilakukan lebih dalam dan tidak sekadar pada *head layers*.

```
1.    model = object_detector.create(train_data,
2.                        model_spec=model_spec.get('efficientdet_lite2'),
3.                        batch_size=30,
4.                        epochs=140,
5.                        train_whole_model=True,
6.                        validation_data=validation_data)}
```

Gambar 3.12 *Hyperparameter* model

3.4.3 Ekspor Model

Setelah melakukan pelatihan model maka hasil pelatihan tersebut dapat diekspor menjadi suatu file dengan format tflite. Format file tersebut dapat dijalankan pada perangkat bergerak. Proses pendeteksian pun dapat dilakukan pada komputer maupun perangkat bergerak. File model yang dihasilkan sebanyak empat buah sesuai dengan jumlah jenis dataset yang dilatih, yakni model dengan dataset optimal, bias1, bias2, dan bias3.

3.5 Evaluasi Model

Evaluasi dilakukan untuk mengetahui performa model secara keseluruhan. Matriks evaluasi yang digunakan adalah *Average Precision.* Matriks tersebut menggunakan nilai *precision* dan *recall* untuk memperoleh suatu nilai hasil evaluasi. Rumus dari AP, *precision, dan recall* dapat dilihat pada persamaan (2.5), (2.6), dan (2.7). Kemudian, pada arsitektur model ini, evaluasi AP menggunakan standar COCO sehingga secara otomatis nilai ambang batas dari *Intersection of Union* sebesar 0,5. Memulai fungsi evaluasi model dapat dilakukan dengan menjalankan baris kode pada Gambar 3.13. Saat kode pada baris pertama dijalankan maka sistem secara otomatis melakukan perhitungan nilai *Average Precision.* Setelah mendapatkan nilai AP dari masing-masing model, peneliti akan mengukur besar pengaruh bias dengan cara menghitung besar selisih antara dataset bias cahaya (dataset Bias1), sudut pengambilan (dataset Bias2), dan pelabelan (dataset Bias3) dengan dataset acuan (dataset Optimal).

Saat proses perhitungan berlangsung, sistem akan melakukan pengujian menggunakan data tes. Data tes ini berisi kumpulan gambar yang tidak diikutkan dalam proses pelatihan dan tentunya di dalam data tes ini sudah ada semua objek yang akan diuji. Proses pembuatan data tes ini memiliki tahapan yang sama dengan pembuatan dataset untuk pelatihan model. Hal yang

berbeda terletak pada jumlah gambar yang digunakan. Pada penelitian ini jumlah gambar yang digunakan sebanyak 61 gambar dengan detail jumlah label dapat dilihat pada Tabel 3.6.

```
1.  model.evaluate_tflite('model.tflite', test_data)
```

Gambar 3.13 Baris Kode untuk menjalankan evaluasi

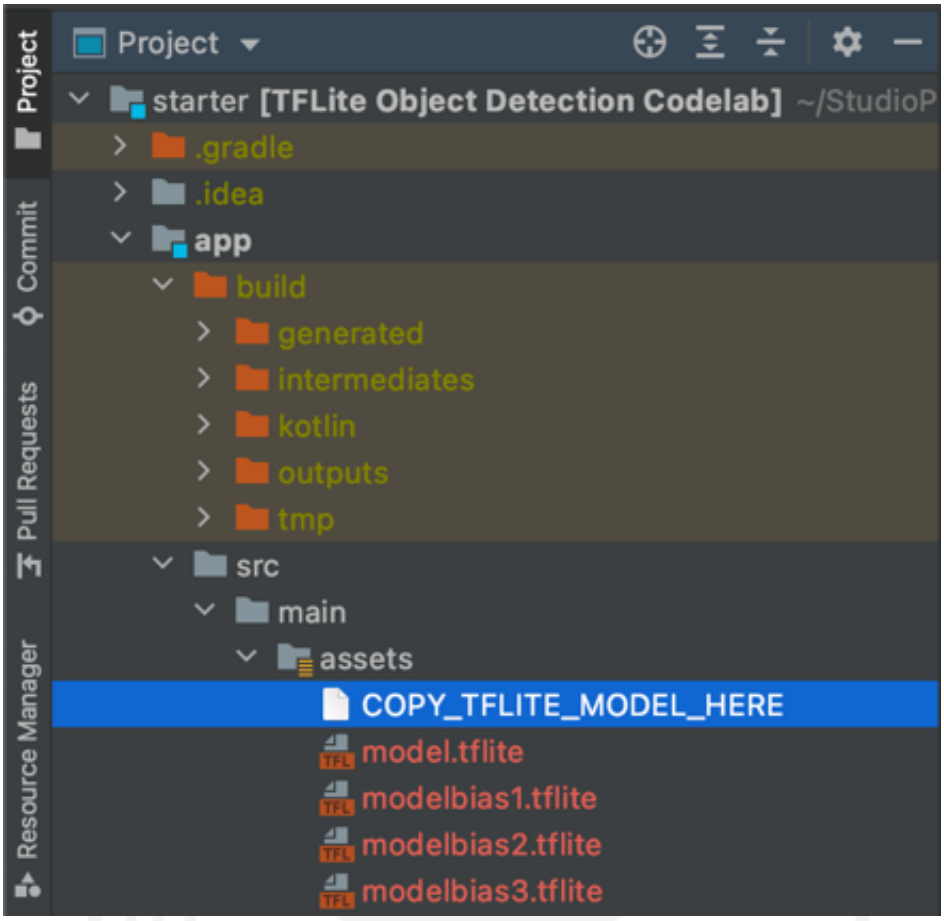
Tabel 3.6 Jumlah label pada data tes

No.	Nama Kelas	Dataset Optimal
1.	NESCAFE Latte 24x220ml ID	23
2.	NESCAFE Cappuccino 24x220ml ID	22
3.	NESCAFE Caramel Macchiato 24x220ml ID	24
4.	Milo NutriUp	29
5.	NESCAFE Original	21
6.	NESCAFE Latte Can	22
7.	NESCAFE Mocha Can	24
8.	Milo Original	30
9.	Milo Calcium	25
10.	NS UHT Black	33
11.	NS UHT CoffeCream	26
12.	NS UHT FrenchVanilla	31
13	White Tea	30
14.	White Malt	32
Total label		371

3.6 Menjalankan Model pada Aplikasi Perangkat Bergerak

Pada tahap ini, peneliti akan membangun sebuah aplikasi menggunakan IDE (*Integreted Development Environment*) yang biasa digunakan untuk membangun sebuah aplikasi. Pada penelitian ini, peneliti akan mebangun aplikasi berbasis android sehingga IDE resmi yang digunakan adalah Android Studio dengan menggunakan bahasa pemrograman Kotlin.

Lalu, peneliti memanfaatkan kode yang telah disediakan oleh *tflite model maker* untuk membangun aplikasi deteksi objek sederhana. Oleh karena itu, tahap awal yang perlu dilakukan adalah mengunduh sumber kode tersebut. Pada kode tersebut terdapat pustaka *tflite model maker* yang berfungsi untuk membaca file hasil ekspor model yang memiliki ekstensi tflite. File hasil ekspor tersebut akan dimasukkan ke dalam aplikasi dengan menyimpannya pada sub folder “assets” dari projek aplikasi ini. Gambar 3.14 menunjukkan detail dari lokasi penyimpanan model yang telah dieksport.



Gambar 3.14 Sub folder penyimpanan model pada projek aplikasi

Kemduian, aplikasi ini memiliki fungsi yang dapat digunakan untuk memanggil kamera karena masukan gambar pada aplikasi ini bersumber dari kamera. Setelah itu, aplikasi akan memproses gambar tersebut untuk melakukan penyesuaian format gambar dan ukuran gambar yang dapat diproses oleh model deteksi objek. Setelah itu, model akan melakukan proses pendeteksian dan akan menampilkan hasilnya pada aplikasi.

3.7 Rancangan Antarmuka Aplikasi

Antarmuka merupakan bagian yang akan langsung berinteraksi dengan pengguna. Gambar 3.15 merupakan tampilan halaman utama aplikasi. Pada halaman ini, aplikasi dapat digunakan untuk mengambil gambar dari kamera utama *smartphone*. Pengambilan gambar dapat dilakukan dengan cara menekan tombol yang telah disediakan. Setelah itu, aplikasi dapat melakukan pengambilan gambar, pemrosesan gambar, dan menampilkan hasil deteksi gambar.

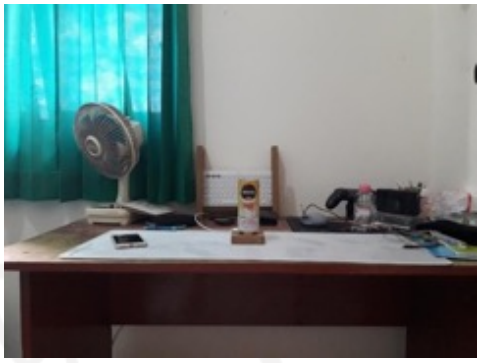
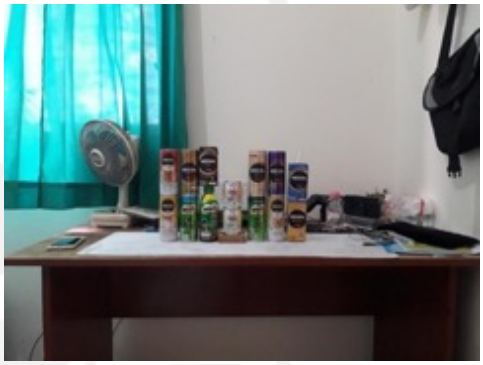


Gambar 3.15 Tampilan rancangan aplikasi

3.8 Pengujian Model pada Perangkat Bergerak

Pada tahapan ini, peneliti akan menguji kemampuan setiap model untuk dapat mendeteksi objek pada jarak 50cm dan 100cm. Pengujian pertama dilakukan dengan cara meletakkan satu objek dan pengujian kedua dilakukan dengan cara meletakkan ke-14 objek untuk dideteksi secara bersamaan, untuk lebih detailnya dapat dilihat pada Tabel 3.7. Pengujian ini menggunakan perangkat Samsung galaxy J3 pro dengan spesifikasi kamera yang memiliki ukuran sensor sebesar 8 Megapixel. Setelah pengujian dilakukan, peneliti akan melakukan analisis hasil dan melakukan analisis pada setiap dataset. Dari hasil analisis tersebut, peneliti akan mencoba untuk menerapkan hasil analisis kemudian melakukan peningkatan performa model untuk dataset optimal.

Tabel 3.7 Contoh pengambilan gambar pada perangkat lunak

	Jarak 50cm	Jarak 100cm
Satu objek		
Empat belas objek		

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Nilai *loss*

Dari proses pemodelan yang telah dilalui, dihasilkan empat buah model yang telah dilatih menggunakan arsitektur model *efficientdet-lite2.* Dalam proses pemodelan, dataset optimal, bias1, bias2, dan bias3 dilatih menggunakan *hyperparameter* yang sama pada Gambar 3.12. Di atas kertas, model Optimal memiliki nilai *loss* paling kecil diantara model yang lainnya dengan menggunakan *hyperparameter* yang sama. Perbandingan nilai *loss* training tiap model dapat lihat pada Tabel 4.1.

Tabel 4.1 Perbandingan *loss training* dan *loss validation*

	Optimal	Bias1	Bias2	Bias3
Loss training	0,2297	0,2476	0,2856	0,2815
Loss validation	0,3474	0,6668	0,5984	0,6470

4.2 Hasil Skor Average Precision (AP)

Keempat model yang telah diekspor menjadi model yang telah dioptimalkan untuk perangkat bergerak dengan *framework tensorflow lite* yang disimpan dengan ekstensi file tflite. Kemudian, model tersebut dievaluasi menggunakan matrik evaluasi AP COCO dengan menggunakan data tes khusus yang telah disiapkan. Hasil evaluasi ini menghasilkan angka yang dapat dijadikan sebagai tolak ukur performa tiap model dan tiap kelas. Pada model *efficientdet-lite2,* model tersebut memiliki nilai rata-rata AP sebesar 0,3397 (Tomas, 2021). Suatu model akan memiliki performa yang dikatan baik ketika melebihi nilai rata-rata dari model yang digunakan. Kemudian untuk menentukan nilai pengaruh bias, nilai AP dari model dengan dataset Optimal akan dijadikan sebagai *base line model*.

Tabel 4.2 Perbandingan nilai AP model

	Optimal	Bias1	Bias2	Bias3
AP	0,6211	0,4902	0,5215	0,122

Tabel 4.2 menunjukkan perbandingan hasil evaluasi tiap model secara keseluruhan. Dari nilai tersebut, model dengan dataset Bias3 memiliki nilai yang paling rendah dan dapat dikatakan memiliki performa yang tidak baik karena nilai AP dari model dengan dataset Bias3

lebih rendah dari nilai rata-rata AP *efficientdet-lite2*. Kemudian model yang dijadikan sebagai acuan pengaruh bias atau *bas line model*, yakni model dengan dataset Optimal. Model dengan dataset tersebut memperoleh niali AP sebesar 0,6211. Nilai tersebut sudah lebih besar dari nilai rata-rata model yang digunakan. Kemudian, untuk model dengan dataset Bias1, Bias2, dan Bias3 masing-masing mengalami penurunan nilai AP dari *baseline model* sebesar 36,9%, 16%, dan 98%. Penurunan nilai tersebut menunjukkan bahwa bias dalam dataset mempengaruhi performa deteksi objek yang ditunjukkan dengan penurunan nilai AP dari nilai *baseline* model.

Tabel 4.3 Perbandingan nilai AP tiap kelas

No.	Nama Kelas	Optimal	Bias1	Bias2	Bias3
1.	NESCAFE Latte 24x220ml ID	0,7241	0,4093	0,6627	0,0338
2.	NESCAFE Cappuccino 24x220ml ID	0,7389	0,6863	0,6226	0,0845
3.	NESCAFE Caramel Macchiato 24x220ml ID	0,5630	0,3702	0,6414	0,0022
4.	Milo NutriUp	0,7334	0,5961	0,6068	0,1813
5.	NESCAFE Original	0,5339	0,5505	0,3798	0,1741
6.	NESCAFE Latte Can	0,4995	0,5021	0,3332	0,1007
7.	NESCAFE Mocha Can	0,5584	0,5546	0,4813	0,1477
8.	Milo Original	0,7084	0,4502	0,6023	0,0313
9.	Milo Calcium	0,6252	0,5722	0,7361	0,0839
10.	NS UHT Black	0,5430	0,4470	0,1489	0,2991
11.	NS UHT CoffeCream	0,6712	0,4777	0,6981	0,1392
12.	NS UHT FrenchVanilla	0,5936	0,3528	0,7562	0,2934
13.	White Tea	0,6485	0,4106	0,5251	0,0795
14.	White Malt	0,5632	0,4879	0,6280	0,1153

Kemudian pada Tabel 4.3 menunjukkan nilai AP yang didapatkan tiap kelas. Pada model Bias3 dengan nomor kelas 10, kelas tersebut memiliki nilai AP paling rendah. Penerapan bias terhadap kelas tersebut memiliki pengaruh yang sangat signifikan hingga menghasilkan nilai AP sebesar 0,1489.

Antara model Optimal dengan model Bias2, pada setiap kelasnya memiliki nilai AP yang tidak berbeda jauh. Perbedaan yang cukup signifikan terdapat pada kelas nomor 5,6,7, dan 14. Kemudian, model Bias1 dan Bias3 dibandingkan dengan model Bias2 dan model Optimal memiliki perbedaan nilai yang sangat signifikan, kecuali pada model Bias2 dengan nomor kelas 1,2,3 dan 4 tidak memiliki perbedaan yang cukup jauh karena keempat produk tersebut memiliki kemasan yang tidak memantulkan cahaya sehingga pengaruh bias cahaya tidak terlalu berpengaruh. Pada model Bias3, seluruh kelasnya memiliki nilai yang rendah dan dapat dipastikan pengaruh kesalahan pemberian label sebesar 50% di setiap kelas dapat menurunkan performa deteksi objek.

4.3 Implementasi Model pada Perangkat Bergerak

Peneliti menggunakan model yang telah diekspor untuk perangkat bergerak dengan ekstensi file tflite. File tersebut dapat digunakan dalam aplikasi bergerak. Pada kasus ini, peneliti membangun aplikasi pada sistem operasi android. Oleh karena itu, pembuatan kode dilakukan menggunakan *tools Android Studio* dengan menggunakan bahasa Kotlin. Pembuatan aplikasi ini memanfaatkan *library* yang telah disediakan oleh tensorflow serta menggunakan sumber kode dari *tflite model maker*. Berikut merupakan langkah-langkah yang diperlukan untuk membangun aplikasi serta proses pendeteksian:

a. Pertama-tama yakni, mengunduh *source code* aplikasi android yang telah disediakan oleh *tflite model maker*. Dalam aplikasi ini, tombol utama untuk dapat memperoleh hasil deteksi diinisiasi dengan menekan tombol ambil gambar atau "*take photo"*. Setelah tombol tersebut ditekan maka secara otomatis kode pada baris ke 3 pada Gambar 4.1 akan memanggil fungsi yang dapat membuka kamera untuk mengambil gambar. Baris kode tersebut akan memanggil fungsi yang ada pada Gambar 4.2. Baris kode ke-4 s.d. ke-7 merupakan baris kode untuk menangani *error*.

```
1. override fun onClick(v: View?) {
2.     when (v?.id) {
3.         R.id.captureImageFab -> {
4.             try {
5.                 dispatchTakePictureIntent()
6.             } catch (e: ActivityNotFoundException) {
7.                 Log.e(TAG, e.message.toString())
8.             }
9.         }
10.       }
11. }
```

Gambar 4.1 Kode untuk inisiasi pemanggilan fungsi memanggil gambar

b. Setelah itu, digunakan kode pada Gambar 4.2 untuk membuka kamera sehingga pengguna dapat mengambil gambar objek yang akan dideteksi. Di dalam fungsi *"dispatchTakePictureIntent()"* pada baris pertama terdapat *Intent* pada baris kedua yang berfungsi untuk membuka kamera. Setelah itu, baris kode ke-4 s.d. ke-18 akan memproses gambar yang telah diambil, proses yang dilakukan berupa mengubah bentuk gambar menjadi format bitmap agar setelah prediksi dilakukan sistem dapat memberikan *bounding box* terhadap objek yang ada pada gambar. Selanjutnya secara otomatis baris kode ke-19 s.d. ke-20 akan memanggil fungsi yang akan melakukan pendeteksian objek.

```
1. private fun dispatchTakePictureIntent() {
2.         Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->
3.             // Ensure that there's a camera activity to handle the intent
4.             takePictureIntent.resolveActivity(packageManager)?.also {
5.                 // Create the File where the photo should go
6.                 val photoFile: File? = try {
7.                     createImageFile()
8.                 } catch (e: IOException) {
9.                     Log.e(TAG, e.message.toString())
10.                        null
11.                    }
12.                    // Continue only if the File was successfully created
13.                    photoFile?.also {
14.                        val photoURI: Uri = FileProvider.getUriForFile(
15.                            this,
16.             "org.tensorflow.codelabs.objectdetection.fileprovider",
17.                            it
18.                        )
19.                      takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
    photoURI)
20.                        startActivityForResult(takePictureIntent,
    REQUEST_IMAGE_CAPTURE)
21.                    }
22.            }
23.        }
24.    }
```

Gambar 4.2 Fungsi untuk membuka kamera dan memproses gambar

c. Setelah pemrosesan gambar, maka aplikasi akan memanggil fungsi deteksi objek pada Gambar 4.3. Pada kode baris pertama terdapat fungsi "*runObjectDetection( )"* yang akan melakukan prediksi menggunakan model yang telah diekspor. Baris ketiga membaca gambar yang sudah diubah kedalam format bitmap. Kemudian, kode pada baris ke-6 s.d. ke-9 menetapkan nilai ambang batas deteksi dan jumlah deteksi maksimal yang dapat dilakukan. Kode pada baris ke-10 s.d. ke-13 akan menggunakan model deteksi objek untuk melakukan pendeteksian. Kemudian, kode pada baris ke-17 melakukan pendeteksian terhadap gambar. Selanjutnya, kode baris ke-20 s.d. ke-26 memproses hasil deteksi yang telah dilakukan dengan cara menghasilkan nilai prediksi dan label objek. Selanjutnya, pada kode baris ke-30 akan memanggil fungsi "*drawDetectionResult"* yang terdapat pada Gambar 4.4. Terakhir, bari ke-32 berfungsi untuk menampilkan hasil prediksi berupa gambar bitmap yang telah diberikan *bounding box,* nilai prediksi, dan label.

```
1. private fun runObjectDetection(bitmap: Bitmap) {
2.     // Step 1: create TFLite's TensorImage object
3.     val image = TensorImage.fromBitmap(bitmap)
4.
5.     // Step 2: Initialize the detector object
6.     val options = ObjectDetector.ObjectDetectorOptions.builder()
7.         .setMaxResults(15)
8.         .setScoreThreshold(0.2f)
9.         .build()
10.       val detector = ObjectDetector.createFromFileAndOptions(
11.           this, // the application context
12.           "modeloptimal1.tflite", // must be same as the filename in
   assets folder
13.           options
14.       )
15.
16.       // Step 3: feed given image to the model and print the detection
   result
17.       val results = detector.detect(image)
18.
19.       // Step 4: Parse the detection result and show it
20.       val resultToDisplay = results.map {
21.           // Get the top-1 category and craft the display text
22.           val category = it.categories.first()
23.           val  text  =   "${category.score.times(100).toInt()}%"  +
   "${category.label}"
24.
25.           // Create a data object to display the detection result
26.           DetectionResult(it.boundingBox, text)
27.       }
28.
29.       // Draw the detection result on the bitmap and show it.
30.       val     imgWithResult     =     drawDetectionResult(bitmap,
   resultToDisplay)
31.       runOnUiThread {
32.           inputImageView.setImageBitmap(imgWithResult)
33.       }
34.   }
```

Gambar 4.3 Fungsi untuk pendeteksian model dan menampilkan hasil deteksi

d. Pada tahapan ini, gambar bitmap akan diberikan *bounding box,* nilai prediksi, dan label objek. Pada baris ke-6 s.d. baris ke-9 berfungsi untuk menentukan area gambar yang akan diproses. Selanjutnya, baris ke-10 akan melakukan perulangan sebanyak jumlah objek yang terdeteksi. Pada baris ke-13 s.d. baris ke-38 dilakukan proses pembuatan *bounding box* dengan warna merah dan menentukan letak *bounding box* tersebut berdasarkan letak objek yang terdeteksi. Kemudian, letak label dan nilai prediksi mengikuti letak dari *bounding box.* Setelah proses ini selesai dilakukan, maka fungsi "*drawDetectionResult"* akan mengembalikan nilai berupa gambar bitmap yang telah diberikan *bounding box*, nilai prediksi, dan label objek. Selanjutnya akan ditampilkan menuju halaman utama aplikasi.

```
1.  private fun drawDetectionResult(
2.      bitmap: Bitmap,
3.      detectionResults: List<DetectionResult>
4.  ): Bitmap {
5.      val outputBitmap = bitmap.copy(Bitmap.Config.ARGB_8888, true)
6.      val canvas = Canvas(outputBitmap)
7.      val pen = Paint()
8.      pen.textAlign = Paint.Align.LEFT
9.
10.     detectionResults.forEach {
11.         // draw bounding box
12.
13.         pen.color = Color.RED
14.         pen.strokeWidth = 8F
15.         pen.style = Paint.Style.STROKE
16.         val box = it.boundingBox
17.         canvas.drawRect(box, pen)
18.
19.
20.         val tagSize = Rect(0, 0, 0, 0)
21.
22.         // calculate the right font size
23.         pen.style = Paint.Style.FILL_AND_STROKE
24.         pen.color = Color.YELLOW
25.         pen.strokeWidth = 2F
26.
27.         pen.textSize = MAX_FONT_SIZE
28.         pen.getTextBounds(it.text, 0, it.text.length, tagSize)
29.         val fontSize: Float = pen.textSize * box.width() /
    tagSize.width()
30.
31.         // adjust the font size so texts are inside the bounding box
32. //            if (fontSize < pen.textSize) pen.textSize = fontSize
33.
34.         var margin = (box.width() - tagSize.width()) / 2.0F
35.         if (margin < 0F) margin = 0F
36.         canvas.drawText(
37.             it.text, box.left + margin,
38.             box.top + tagSize.height().times(1F) + X, pen
39.         )
40.     }
41.     return outputBitmap
42. }
```

Gambar 4.4 Fungsi untuk pembuatan *bounding box,* nilai prediksi, dan label objek

4.4 Tampilan Antarmuka Aplikasi

Gambar 4.6 merupakan tampilan aplikasi dengan prediksinya dalam bentuk aplikasi bergerak yang dirancang dengan android studio. Pembuatan aplikasi termasuk mudah karena tensorflow menyediakan *source code* serta tutorialnya dalam implementasi kode tersebut. Aplikasi ini memanfaatkan hasil eksport file tflite yang digunakan untuk melakukan deteksi objek pada perangkat bergerak. Kemudian, Gambar 4.5 merupakan baris kode yang digunakan

untuk membuat tampilan utama aplikasi. Baris ke-1 sampai baris ke7 merupakan baris kode yang bertugas untuk menampilkan gambar hasil deteksi yang telah dilakukan oleh aplikasi. Selanjutnya, baris ke-8 sampai dengan baris ke-18 merupakan baris yang berfungsi sebagai tombol yang dapat membuka kamera, kemudian setelah mengambil gambar akan otomatis terjadi pemrosesan gambar dan hasilnya akan ditampilkan pada aplikasi.

```
1.   <ImageView
2.          android:id="@+id/imageView"
3.          android:layout_width="match_parent"
4.          android:layout_height="match_parent"
5.          android:adjustViewBounds="true"
6.          android:contentDescription="@null"
7.          android:scaleType="fitCenter" />
8. <Button
9.          android:id="@+id/captureImageFab"
10.            android:layout_width="wrap_content"
11.            android:layout_height="wrap_content"
12.            android:layout_alignParentBottom="true"
13.            android:layout_centerHorizontal="true"
14.            android:layout_gravity="center|bottom"
15.            android:layout_margin="16dp"
16.            android:drawableLeft="@drawable/ic_camera"
17.            android:text="@string/tv_take_photo"
18.            android:textAllCaps="false" />
```

Gambar 4.5 Baris kode untuk tampilan utama aplikasi



Gambar 4.6 Tampilan aplikasi deteksi objek

4.5 Percobaan Model pada Perangkat Bergerak

Percobaan dilakukan menggunakan perangkat Samsung Galaxy J3 Pro. Pengujian dilakukan di dalam ruangan dengan pencahayaan yang cukup.

Pengujian Pertama

Pengujian pertama menggunakan jarak pengambilan foto sejauh 50 cm dan 100 cm dari objek dan objek yang dideteksi hanya satu untuk setiap kelas. Pada Tabel 4.4, hasil yang didapatkan bahwa performa model Bias1 dan Bias3 tidak memiliki performa yang baik, terutama pada model bias3. Pada model Bias3 banyak yang menghasilkan salah deteksi sesuai dengan kondisi bias yang diterapkan pada dataset tersebut. Kemudian, untuk jarak 50 cm model Optimal dan Bias2 memiliki performa yang baik dengan memperoleh nilai yang cukup besar

untuk setiap kelasnya tetapi pada kelas White Tea dan White Malt pada model Optimal masih memiliki nilai akurasi yang rendah. Berdasarkan data yang ada pada kelas tersebut, data dengan jarak pengambilan gambar lebih dari 50cm memiliki jumlah yang lebih banyak dan kondisi gambar dengan jarak tersebut pada setiap gambar memiliki jumlah objek lebih dari tiga sehingga saat dilakukan pengujian dengan satu objek hasil deteksinya tidak stabil. Sedangkan pada model bias2, kelas White Tea dan White malt memiliki performa yang baik karena didalam dataset terdapat gambar objek dengan kriteria pengujian gambar dan memiliki distribusi penyebaran data yang baik.

Performa model Optimal untuk jarak pengujian 100cm memiliki performa yang kurang baik karena masih banyak objek yang tidak dapat terdeteksi dibandingkan dengan model Bias2. Pada dataset model Optimal jumlah data dengan jarak pengambilan gambar sejauh 100cm sangat sedikit sehingga ketika diuji dengan jarak tersebut performa model dalam mendeteksi objek menjadi berkurang hingga tidak dapat memberikan hasil prediksi.

Tabel 4.4 Perbandingan hasil pengujian pada jarak 50 cm dan 100 cm

Kelas	Jarak	Optimal	Bias1	Bias2	Bias3
NESCAFE Latte 24x220ml ID	50 cm	93%	Salah deteksi	82%	75%
	100 cm	35%	Tidak terdeteksi	71%	Tidak terdeteksi
NESCAFE Cappuccino 24x220ml ID	50 cm	92%	89%	83%	61%
	100 cm	Tidak terdeteksi	Tidak terdeteksi	88%	Tidak terdeteksi
NESCAFE Caramel Macchiato 24x220ml ID	50 cm	90%	Salah deteksi	88%	Tidak terdeteksi
	100 cm	46%	58%	84%	Tidak terdeteksi
Milo NutriUp	50 cm	85%	78%	90%	Salah deteksi
	100 cm	20%	48%	57%	Salah deteksi
NESCAFE Original Can	50 cm	52%	Tidak terdeteksi	50%	Salah deteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	54%	Tidak terdeteksi
NESCAFE Latte Can	50 cm	73%	Tidak terdeteksi	89%	Tidak terdeteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
NESCAFE Mocha Can	50 cm	86%	Tidak terdeteksi	88%	Salah deteksi
	100 cm	21%	Tidak terdeteksi	89%	Tidak terdeteksi
Milo Original	50 cm	85%	Salah deteksi	74%	Salah deteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
Milo Calcium	50 cm	91%	79%	87%	Salah deteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	45%	Salah deteksi
NS UHT Black	50 cm	90%	Tidak terdeteksi	Tidak terdeteksi	53%
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
NS UHT CoffeeCream	50 cm	88%	63%	71%	Salah deteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	66%	Tidak terdeteksi
NS UHT FrenchVanilla	50 cm	87%	74%	90%	Tidak terdeteksi
	100 cm	55%	Tidak terdeteksi	83%	Tidak terdeteksi
White Tea	50 cm	40%	Tidak terdeteksi	93%	Tidak terdeteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	37%	Tidak terdeteksi
White Malt	50 cm	34%	81%	76%	Tidak terdeteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	41%	Tidak terdeteksi

Pengujian Kedua

Pada pengujian kedua, jarak pengambilan gambar masih sama, yakni 50cm dan 100cm. Hal berbeda pada pengujian ini adalah jumlah objek yang akan dideteksi pada gambar. Selain itu, semua objek berada pada satu gambar yang sama. Hasil pengujian ditunjukkan pada Tabel 4.5. Pada model Optimal dengan jarak 50cm mampu mendeteksi sembilan objek dengan benar dan pada jarak 100cm dapat mendeteksi tujuh objek. Pada model Bias2 berhasil mendeteksi sembilam objek pada jarak 50cm dan tujuh objek pada jarak 100cm. Performa dari model Optimal dan Bias2 pada pengujian ini masih menunjukkan hasil yang baik dengan memberikan nilai akurasi diatas 50% dan pada model Bias1 dan Bias3 memiliki performa yang tidak baik karena jumlah objek yang dideteksi hanya tiga dan masih banyak objek yang tidak dapat terdeteksi.

Tabel 4.5 Pengujian empat belas objek secara bersamaan

Kelas	Jarak	Optimal	Bias 1	Bias 2	Bias 3
NESCAFE Latte 24x220ml ID	50 cm	55%	65%	Tidak terdeteksi	20%
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
NESCAFE Cappuccino 24x220ml ID	50 cm	77%	53%	59%	Tidak terdeteksi
	100 cm	60%	Tidak terdeteksi	61%	34%
NESCAFE Caramel Macchiato 24x220ml ID	50 cm	44%	Tidak terdeteksi	42%	Tidak terdeteksi
	100 cm	36%	Salah deteksi	34%	69%
Milo NutriUp	50 cm	81%	Tidak terdeteksi	55%	34%
	100 cm	35%	Tidak terdeteksi	32%	34%
NESCAFE Original Can	50 cm	33%	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
	100 cm	33%	Tidak terdeteksi	Salah deteksi	Salah deteksi
NESCAFE Latte Can	50 cm	21%	Tidak terdeteksi	34%	Tidak terdeteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	62%	Tidak terdeteksi
NESCAFE Mocha Can	50 cm	75%	Tidak terdeteksi	61%	Tidak terdeteksi
	100 cm	25%	Tidak terdeteksi	50%	Tidak terdeteksi
Milo Original	50 cm	Tidak terdeteksi	Tidak terdeteksi	24%	Salah deteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
Milo Calcium	50 cm	Tidak terdeteksi	Tidak terdeteksi	78%	Salah deteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	74%	Tidak terdeteksi
NS UHT Black	50 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
NS UHT CoffeeCream	50 cm	35%	Tidak terdeteksi	83%	Tidak terdeteksi
	100 cm	26%	Tidak terdeteksi	72%	Tidak terdeteksi
NS UHT FrenchVanilla	50 cm	65%	Tidak terdeteksi	52%	Salah deteksi
	100 cm	65%	Tidak terdeteksi	86%	Tidak terdeteksi
White Tea	50 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
White Malt	50 cm	27%	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi
	100 cm	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi	Tidak terdeteksi

Dari hasil pengujian tersebut, peneliti melakukan analisis terhadap kondisi dataset dari setiap kelas yang ada pada setiap model. Model Optimal dan Bias2 memiliki performa yang baik terutama pada ketiga kelas berikut, NESCAFE Latte 24x220ml ID, NESCAFE Cappuccino 24x220ml ID, dan NESCAFE Caramel Macchiato 24x220ml ID. Ketiga kelas tersebut memiliki kuatias pencahayaan yang baik, kondisi pencahayaan yang konsisten, tidak ada gambar yang buram, warna objek pada gambar konsisten dan tidak berubah secara

signifikan, serta pada setiap gambar yang terdapat objek tersebut diberikan label dan tidak ada yang terlewati. Sedangkan sebelas kelas lainnya memiliki kondisi gambar yang dominan gelap, cukup buram, dan masih ada objek yang belum diberikan label pada beberapa gambar. Kemudian, peneliti mencoba untuk melakukan pelabelan terhadap objek yang belum diberikan label. Dataset yang digunakan oleh peneliti, yakni dataset untuk model Optimal.

Tabel 4.6 Perbandingan jumlah label sebelum dan sesudah perbaikan pada dataset Optimal

No.	Nama Kelas	Sebelum	Sesudah
1.	NESCAFE Latte 24x220ml ID	134	134
2.	NESCAFE Cappuccino 24x220ml ID	110	110
3.	NESCAFE Caramel Macchiato 24x220ml ID	123	123
4.	Milo NutriUp	253	393
5.	NESCAFE Original	180	180
6.	NESCAFE Latte Can	198	198
7.	NESCAFE Mocha Can	185	185
8.	Milo Original	186	426
9.	Milo Calcium	235	335
10.	NS UHT Black	270	270
11.	NS UHT CoffeCream	167	267
12.	NS UHT FrenchVanilla	123	273
13	White Tea	243	443
14.	White Malt	146	346
Total label		2543	3683

Tabel 4.9 menunjukkan nilai akurasi setiap objek. Pengujian dengan jarak 50cm memiliki hasil yang sangat baik karena semua kelas masih dapat mendeteksi objek dengan nilai akurasi yang cukup besar dibandingkan dengan model Optimal yang belum diperbaiki. Kemudian, untuk pengujian jarak pada 100cm mengalami peningkatan karena objek yang sebelumnya tidak dapat terdeteksi menjadi berhasil dideteksi meskipun masih ada beberapa objek yang memiliki nilai akurasi rendah. Saat dilakukan pengecekan terhadap dataset, jumlah gambar dengan jarak 100cm memiliki jumlah yang sedikit karena kondisi data di lapangan yang tidak memungkinakan untuk mengambil gambar dengan jarak tersebut. Selain itu, masih terdapat gambar yang cukup buram pada jarak 100cm.

Tabel 4.7 Nilai AP sebelum dan sesudah perbaikan pada model dataset Optimal

	Sebelum	Sesudah
AP	0,6211	0,6703

Tabel 4.8 Nilai AP setiap kelas sebelum dan sesudah perbaikan dataset Optimal

No.	Nama Kelas	Sebelum	Sesudah
1.	NESCAFE Latte 24x220ml ID	0,7241	0,6941
2.	NESCAFE Cappuccino 24x220ml ID	0,7389	0,7140
3.	NESCAFE Caramel Macchiato 24x220ml ID	0,5630	0,5239
4.	Milo NutriUp	0,7334	0,7056
5.	NESCAFE Original	0,5339	0,7133
6.	NESCAFE Latte Can	0,4995	0,6591
7.	NESCAFE Mocha Can	0,5584	0,6680
8.	Milo Original	0,7084	0,6173
9.	Milo Calcium	0,6252	0,5983
10.	NS UHT Black	0,5430	0,7270
11.	NS UHT CoffeCream	0,6712	0,7853
12.	NS UHT FrenchVanilla	0,5936	0,7857
13.	White Tea	0,6485	0,6837
14.	White Malt	0,5632	0,5746

Tabel 4.7 dan Tabel 4.8 menunjukkan nilai AP untuk keselurusan model dan setiap objek. Peningkatan nilai AP tidak begitu besar tetapi dengan penambahan label tersebut dapat meningkatkan nilai AP dari model dan juga meningkatkan kemampuan deteksi objek menjadi lebih baik yang dapat dilihat juga pada tabel Tabel 4.9.

Tabel 4.9 Hasil Pengujian dengan dataset optimal sebelum dan sesudah diperbaiki

Kelas	Jarak	Sebelum	Sesudah
NESCAFE Latte 24x220ml ID	50 cm	93%	97%
	100 cm	35%	Tidak terdeteksi
NESCAFE Cappuccino 24x220ml ID	50 cm	92%	95%
	100 cm	Tidak terdeteksi	60%
NESCAFE Caramel Macchiato 24x220ml ID	50 cm	90%	86%
	100 cm	46%	77%
Milo NutriUp	50 cm	85%	97%
	100 cm	20%	82%
NESCAFE Original Can	50 cm	52%	95%
	100 cm	Tidak terdeteksi	29%
NESCAFE Latte Can	50 cm	73%	95%
	100 cm	Tidak terdeteksi	23%
NESCAFE Mocha Can	50 cm	86%	97%
	100 cm	21%	71%
Milo Original	50 cm	85%	87%
	100 cm	Tidak terdeteksi	77%
Milo Calcium	50 cm	91%	93%
	100 cm	Tidak terdeteksi	68%
NS UHT Black	50 cm	90%	93%
	100 cm	Tidak terdeteksi	70%
NS UHT CoffeeCream	50 cm	88%	76%
	100 cm	Tidak terdeteksi	54%
NS UHT FrenchVanilla	50 cm	87%	94%
	100 cm	55%	62%
White Tea	50 cm	40%	79%
	100 cm	Tidak terdeteksi	89%
White Malt	50 cm	34%	91%
	100 cm	Tidak terdeteksi	54%

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil akhir dari pengujian performa deteksi objek, dapat ditarik beberapa kesimpulan. Kesimpulan tersebut adalah sebagai berikut:

a. Model *Efficientdet-lite* dipilih karena model tersebut telah dioptimalkan untuk dapat berjalan pada arsitektur perangkat bergerak.
b. Model dengan dataset Optimal, Bias1, dan Bias2 memiliki nilai yang baik karena melebihi nilai AP dari *baseline model* yang digunakan sedangkan model dengan dataset Bias3 memiliki nilai yang rendah dan lebih kecil dari nilai AP *baseline model*.
c. Model dengan dataset Optimal dijadikan sebagai acuan untuk melihat pengaruh bias pada model dengan dataset yang berbeda. Model ini menghasilkan nilai AP sebesar 0,6211
d. Pemberian bias cahaya, bias sudut pengambilan, dan bias pelabelan masing-masing menghasilkan nilai AP sebesar 0,3918, 0,5215, dan 0,122.
e. Selisih nilai AP dataset bias dengan nilai AP dataset acuan, dihasilkan penuruan nilai AP sebesar 36,9%, 16%, dan 98%. Model dengan dataset bias pelabelan harus dihindari dan diminimalisir karena dapat menurunkan performa deteksi objek sebesar 98% sedangkan bias sudut pengambilan dan bias cahaya masih bisa dimanfaatkan untuk dijadikan dataset karena memiliki pengaruh yang masih kecil yakni sebesar 36,9% dan 16% dengan catatan jumlah data dengan bias tersebut harus dibawah 50% dari jumlah dataset.

5.2 Saran

Dalam penelitian ini masih terdapat kekurangan karena performa deteksi objek yang belum stabil dapat mendeteksi objek dalam kondisi apapun. Oleh karena itu, peneliti berharap penelitian ini bisa terus dikembangkan oleh peneliti-peneliti di masa mendatang yang dapat menerapkan beberapa saran dari penulis seperti:

a. Menambah jumlah dataset dengan variasi data yang lebih beragam.
b. Melakukan penyeimbangan terhadap jumlah variasi dataset.
c. Melakukan penelitian dengan model yang lebih besar.
d. Melakukan pencarian solusi terhadap penggunaan memori yang besar terhadap model *efficientdet-lite*

DAFTAR PUSTAKA

Cahyo, P. E., Rolly, A. M., & Andarsyah, R. (2020). *Tutorial Object Detection People With Faster region-Based Convolutional Neural Network(Faster R-CNN)*. Kreatif.

de Tournemire, P., Nitti, D., Perot, E., Migliore, D., & Sironi, A. (2020). *A Large Scale Event-based Detection Dataset for Automotive*. http://arxiv.org/abs/2001.08499

Deng, Y. (2019). *Deep learning on mobile devices: a review*. *April*, 11. https://doi.org/10.1117/12.2518469

Derman, E. (n.d.). *Dataset Bias Mitigation Through Analysis of CNN Training Scores*.

Harani, N. H., & Hasanah, M. (2020). *Deteksi Objek dan Pengenalan Karakter Plat Nomor Kendaraan Indonesia Berbasis Python* (S. M. Yusril, Ed.). Kreatif Industri Nusantara.

Khosla, A., Zhou, T., Malisiewicz, T., Efros, A. A., & Torralba, A. (2012). Undoing the damage of dataset bias. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7572 LNCS*(PART 1), 158–171. https://doi.org/10.1007/978-3-642-33718-5_12

Kortylewski, A., Egger, B., Morel-Forster, A., Schneider, A., Gerig, T., Blumer, C., Reyneke, C., & Vetter, T. (2018). *Can Synthetic Faces Undo the Damage of Dataset Bias to Face Recognition and Facial Landmark Detection?* http://arxiv.org/abs/1811.08565

Kortylewski, A., Egger, B., Schneider, A., Gerig, T., Morel-Forster, A., & Vetter, T. (2018). Empirically analyzing the effect of dataset biases on deep face recognition systems. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2018-June*, 2174–2183. https://doi.org/10.1109/CVPRW.2018.00283

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). *Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone*. 4–6. https://doi.org/10.1111/mice.12387

Manajang, D., Dompie, S., & Jacobus, A. (2020). Implementasi Framework Tensorflow Object Detection Dalam Mengklasifikasi Jenis Kendaraan Bermotor. *Jurnal Teknik Informatika*, *15*(3), 171–178.

McCarthy. (2007). *Artificial Intelligence*. Binus University.

Munir. (2004). *Pengolahan Citra Digital*. Informatika.

Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., & Sun, J. (2019). Objects365: A large-scale, high-quality dataset for object detection. *Proceedings of the IEEE*

International Conference on Computer Vision, *2019-Octob*, 8429–8438. https://doi.org/10.1109/ICCV.2019.00852

Spagnolo, P., Filieri, F., Distante, C., Mazzeo, P. L., & D'Ambrosio, P. (2019). A new annotated dataset for boat detection and re-identification. *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2019*, *September*. https://doi.org/10.1109/AVSS.2019.8909831

Tan, M., Pang, R., & Le, Q. v. (2020). EfficientDet: Scalable and efficient object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 10778–10787. https://doi.org/10.1109/CVPR42600.2020.01079

Tomas, R. (2021). *Fast object detection on mobile platforms using neural networks*.

Tommasi, T., Patricia, N., Caputo, B., & Tuytelaars, T. (2017). A deeper look at dataset bias. *Advances in Computer Vision and Pattern Recognition*, *9783319583464*, 37–55. https://doi.org/10.1007/978-3-319-58347-1_2

Tourani, A., Soroori, S., Shahbahrami, A., & Akoushideh, A. (2021). *Iranis: A Large-scale Dataset of Farsi License Plate Characters*. http://arxiv.org/abs/2101.00295

Xia, G., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., & Zhang, L. (n.d.). *DOTA: A Large-Scale Dataset for Object Detection in Aerial Images | Enhanced Reader*. 3974–3983. moz-extension://bd0c5e16-8147-45c5-ae8b-8401aaca2c83/enhanced-reader.html?openApp&pdf=http%3A%2F%2Fopenaccess.thecvf.com%2Fcontent_cvpr_2018%2Fpapers%2FXia_DOTA_A_Large-Scale_CVPR_2018_paper.pdf

Zheng, S., Hadi Kiapour, M., Yang, F., & Piramuthu, R. (2018). ModaNet: A large-scale street fashion dataset with polygon annotations. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, 1670–1678. https://doi.org/10.1145/3240508.3240652

Srivastava, N., Hinton, G, and Kriszhevsky, A. (2014). Dropout : A Simple Way to Prevent Neural Network . Journal Conference Learning Research, 1929- 1958.

Pham, D. (1994). Neural Network for Chemical Engineers. Amsterdam: Elsevier Press.

LAMPIRAN