PERBANDINGAN PERFORMA SSD MOBILENET V3 LARGE DAN SSD MOBILENET V2 FPNLITE UNTUK DETEKSI OBJEK PADA PRODUK RETAIL

Disusun Oleh:

N a m a : Ahmad Azzam Alhanafi

NIM : 20523033

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PERBANDINGAN PERFORMA SSD MOBILENET V3 LARGE DAN SSD MOBILENET V2 FPNLITE UNTUK DETEKSI OBJEK PADA PRODUK RETAIL

TUGAS AKHIR

Disusun Oleh:

N a m a : Ahmad Azzam Alhanafi

NIM : 20523033

Yogyakarta, 13 Januari 2025

Pembimbing,

(Arrie Kurniawardhani, S.Si., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PERBANDINGAN PERFORMA SSD MOBILENET V3 LARGE DAN SSD MOBILENET V2 FPNLITE UNTUK DETEKSI OBJEK PADA PRODUK RETAIL

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, 13 Januari 2025

Tim Penguji

Arrie Kurniawardhani, S.Si., M.Kom. _____

Anggota 1

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

Anggota 2

Dr. Feri Wijayanto, S.T., M.T. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ahmad Azzam Alhanafi

NIM : 20523033

Tugas akhir dengan judul:

PERBANDINGAN PERFORMA SSD MOBILENET V3 LARGE DAN SSD MOBILENET V2 FPNLITE UNTUK DETEKSI OBJEK PADA PRODUK RETAIL

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, 31 Desember 2024

(Ahmad Azzam Alhanafi)

HALAMAN PERSEMBAHAN

Pada halaman persembahan ini, saya mengucapkan terima kasih sebesar-besarnya kepada Allah Subhanahu Wa Ta'ala yang telah memberikan saya kesempatan menempuh studi Informatika di Universitas Islam Indonesia. Saya juga mengucapkan terima kasih kepada kedua orang tua saya yang telah mengizinkan saya untuk belajar di Universitas Islam Indonesia. Terkahir, tak lupa saya ucapkan terima kasih kepada seluruh dosen dan teman-teman di Universitas Islam Indonesia yang telah memberikan saya inspirasi sehingga bisa menjadi pribadi yang lebih baik kedepannya.

HALAMAN MOTO

“Do not lose hope, nor be sad”

– Quran 3:39

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur kepada Allah Subhanahu Wa Ta'ala karena atas rahmat-Nya lah penulis dapat menyelesaikan laporan tugas akhir yang berjudul "PERBANDINGAN PERFORMA SSD MOBILENET V3 LARGE DAN SSD MOBILENET V2 FPNLITE UNTUK DETEKSI OBJEK PADA PRODUK RETAIL".

Tujuan dari penulisan laporan ini adalah sebagai syarat untuk menyelesaikan program sarjana pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta.

Selama proses penyususan laporan tugas akhir ini, penulis mendapatkan berbagai tantangan baik dari dalam maupun luar kendali penulis. Namun, dengan izin Allah, penulis dapat menyelesaikan laporan tugas akhir ini. Penulis mengucapkan ucapan terikan kasih kepada berbagai pihak, yakni:

1. Allah Subhanahu Wa Ta'ala yang telah memberikan rahmat serta karunianya sehingga penulis dapat menulis laporan tugas akhir ini.
2. Nabi Muhammad Shalallahu Alaihi Wa Sallam, beserta seluruh Rasul dan Nabi-Nabi terdahulu, yang telah memberikan tuntunan hidup yang baik sesuai ajaran Islam.
3. Bapak Ayi Rukandi dan Ibu Lia Amalia, selaku orang tua penulis yang selalu memberikan dukungan kepada penulis terhadap berbagai hal dalam kehidupan.
4. Fatma, Amira, Rasyida, dan Bilqis selaku adik-adik penulis yang telah menemani kehidupan penulis sebagai mahasiswa dan menjadi teman bercerita.
5. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia yang selalu memberikan inspirasi.
6. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika yang telah memberikan ilmu yang sangat berharga kepada penulis sejak semester pertama.
7. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika Program Sarjana, yang telah memberikan penulis pengalaman berharga khususnya dalam bidang sains data.
8. Ibu Arrie Kurniawardhani, S.Si. M.Kom., selaku Dosen Pembimbing Tugas Akhir, yang telah meluangkan waktu dan pikirannya untuk memberikan bimbingan dan arahan khususnya dalam penelitian dan penulisan Tugas Akhir ini.

9. Bapak Rahadian Kurniawan, S.Kom., M.Kom., selaku Dosen Pembina Akademik, yang telah mendukung proses studi penulis serta berbagi pengalaman seputar dunia informatika.
10. Seluruh Dosen dan Civitas Akademika di lingkungan Universitas Islam Indonesia yang telah memberikan penulis pengalaman yang sangat berharga dalam menuntut ilmu di kampus.
11. Seluruh rekan saya di organisasi kampus, seperti LDK Al-Fath UII, CENTRIS FTI UII, Asisten Laboratorium Informatika, dan berbagai organisasi lainnya, yang telah memberikan penulis pelajaran dalam menghadapi dinamika organsisasi sehingga membentuk mental penulis menjadi lebih dewasa.
12. Seluruh pihak yang tidak bisa disebutkan satu per satu, yang telah memberikan dukungan kepada penulis sehingga penelitian dan penulisan laporan tugas akhir ini dapat diselesaikan.

Semoga segala usaha yang dilakukan dalam penelitian ini bernilai pahala di sisi Allah Subhanahu Wa Ta'ala. Penulis menyadari bahwa masih banyak kekurangan yang ada dalam laporan tugas akhir ini. Oleh sebab itu, segala komentar dansaran dibutuhkan untuk kedepannya. Akhir kata, semoga laporan tugas akhir ini memberikan manfaat yang sebesar-besarnya kepada semua pihak.

Yogyakarta, 31 Desember 2024

(Ahmad Azzam Alhanafi)

SARI

Era revolusi industri 4.0 menuntut berbagai sektor industri untuk bergerak lebih cepat, tidak terkecuali industri retail. Sebagai salah satu aspek dalam industri retail yang sangat penting, ketersediaan barang pada rak (On-Shelf Availability) menjadi perhatian khusus karena sangat mempengaruhi loyalitas customer terhadap suatu produk. On-Shelft Availability (OSA) merupakan salah satu aspek yang dapat ditingkatkan performanya dengan memanfaatkan teknologi kecerdasan bautan. Salah satu jenis kecerdasan buatan atau machine learning yang dapat diimplementasikan untuk OSA adalah deteksi objek. Deteksi objek adalah teknologi yang memanfaatkan kemampuan komputer untuk mendeteksi objek yang diinginkan dari gambar atau video yang disediakan. Pada kontek OSA, deteksi objek digunakan untuk mendeteksi ada atau tidaknya produk atau Stock Keeping Unit (SKU) yang dicari.

TensorFlow merupakan framework machine learning yang menyediakan berbagai aristektur yang ditawarkan pada TensorFlow Object Detection API. Salah satu arsitektur deteksi objek yang paling cocok digunakan di perangkat bergerak adalah SSD MobileNet. Pada penelitian sebelumnya, peneliti sudah mengimplementasikan SSD MobileNet pada perangkat bergerak dengan framework TensorFlow 1 dan arsitektur SSD MobileNet. Saat ini, TensorFlow sudah merilis TensorFlow versi 2, sehingga dimungkinkan adanya pembaruan untuk penelitian penulis. Oleh sebab itu, penulis melakukan penelitian terkait perbandingan hasil deteksi objek model SSD MobileNet di TensorFlow 1 dan 2 pada konteks OSA. Tahapan penelitian dimulai dari pengumpulan data, pelabelan data, pemrosesan data, pelatihan data, integrasi model, dan evaluasi model. Pada penelitian ini, penulis menggunakan arsitektur SSD MobileNetV2 untuk TensorFlow 2 dengan tiga variasi input resizer yakni 320x320, 640x640, dan 1024x1024. Untuk TensorFlow 1, input resizer yang digunakan adalah 1024x1024 yang diambil dari pengalaman proyek yang pernah dilakukan oleh penulis.

Hasil penelitian menunjukkan bahwa hasil deteksi objek menggunakan TensorFlow 2 lebih unggul dibandingkan TensorFlow 1 jika deteksi dilakukan menggunakan model asli di perangkat komputer. Hal ini ditunjukkan dengan hasil mAP ketiga model TensorFlow 2 yang lebih tinggi. Namun, saat dikonversi ke TensorFlow Lite, seluruh model baik dari TensorFlow 2 maupun TensorFlow 1 mendapatkan hasil mAP yang tidak jauh berbeda. Waktu inferensi juga menunjukkan perbedaan antar satu model dengan model lainnya. Model yang memiliki input resizer lebih besar cenderung memiliki inference time yang lebih lama.

Kata kunci: On-Shelf Availability, Deteksi Objek, TensorFlow, SSD MobileNet.

GLOSARIUM

Batch Size	Satuan yang digunakan untuk menghitung seberapa banyak gambar yang dilatih dalam satu step.
Kuantisasi	Proses mengubah suatu model machine learning menjadi model yang lebih ringan.
mAP	Metriks evaluasi yang umum digunakna untuk menghitung performa model deteksi objek.
MobileNet	Arsitektur Klasifikasi Gambar berbasis Deep Learning yang memungkinkan model untuk membedakan klasifikasi gambar satu dengan gambar lainnya.
SSD	Arsitektur Deteksi Objek berbasis Deep Learning yang memungkinkan model untuk mendeteksi posisi objek dengan bounding box.
Step	Satuan yang digunakan untuk menghitung satu kali proses update seluruh parameter dalam suatu model yang sedang dilatih.
TensorFlow	Framework Deep Learning yang dirilis oleh Google untuk memudahkan para pengembang Deep Learning untuk mengembangkan model Deep Learning secara End-to-End.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Berkembangnya industri 4.0 atau yang dikenal dengan revolusi industri keempat menjanjikan berbagai kemajuan teknologi dan juga tantangan baru, konsep kecerdasan buatan memiliki peran penting dari transformasi ini seiring dengan meningkatnya volume data dan algoritma pemrosesan (Sanchez, Romero, & Morales, 2021). Retail merupakan salah satu industri di mana keberhasilan implementasi kecerdasan buatan terus bertambah secara konstan (Anica-Popa, Anica-Popa, Rădulescu, & Vrîncianu, 2020).

Dalam industri retail, ketersediaan barang pada rak atau yang dikenal dengan *On-Shelf Availability* (OSA) merupakan indikator performa utama atau *Key Performance Indicator* (KPI) yang sangat berpengaruh terhadap profit dan loyalitas pelanggan. OSA adalah ukuran dari jumlah produk yang tersedia saat pelanggan ingin membeli produk tersebut. Lawan dari OSA adalah *Out-of-Stock* (OOS), yaitu kondisi saat tidak tersedianya suatu produk saat pelanggan ingin membeli produk tersebut (Spielmaker, 2012). Ketika kondisi OOS terjadi, reaksi yang dilakukan oleh pelanggan yakni 31% membeli produk yang mereka butuhkan namun dari tempat lain, 26% membeli merek lain, 19% tetap membeli merek yang sama namun dengan varian/ukuran/rasa yang berbeda, 15% membeli produk tersebut di lain waktu, dan 9% tidak membeli apapun (Gruen, Corsten, & Bharadwaj, 2002).

Kerugian yang dihasilkan dari 9% pelanggan yang tidak membeli apapun mencapai 4 Miliar Euro (Hausruckinger, 2006). Jika kondisi OOS terjadi secara terus menerus, pelanggan terpaksa berpindah toko secara permanen. Persaingan yang masif membuat para pelaku industri ini bekerja keras untuk meningkatkan performa OSA agar para pelanggan puas dan kembali ke toko atau produk mereka. Selama lebih dari 40 tahun penelitian yang dilakukan terhadap OSA, proses yang didapatkan masih sedikit sehingga penelitian dalam bidang ini dianggap belum optimal (Spielmaker, 2012).

Proses audit manual pada rak di industri retail terbukti memakan waktu dan tidak akurat dengan tingkat kesalahan hingga 20% berdasarkan studi dari Universitas Stanford (Chong, Bustan, & Wee, 2016). Namun beberapa tahun terakhir, kemajuan teknologi kecerdasan buatan dan pembelajaran mesin memberikan kontribusi yang besar terhadap perkembangan pengenalan gambar dan deteksi objek pada industri retail. Metode pengenalan gambar dan

deteksi objek dapat membantu pelaku industri dalam melakukan pemeriksaan ketersediaan barang pada rak di toko dan mendapatkan hasil yang konsisten sehingga mereka dapat membuat keputusan bisnis berdasarkan data dengan lebih percaya diri (Singh, 2020).

Saat ini, ada banyak sekali arsitektur deteksi objek yang dapat diterapkan pada industri retail. Beberapa contohnya antara lain Faster R-CNN, Cascade R-CNN, SSD, RetinaNet, RefineDet, CenterNet, dan FCOS (Cai, Wen, Zhang, Du, & Wang, 2021). Selain itu, tersedia pula sistem kerangka kerja yang dapat mengimplementasikan berbagai arsitektur deteksi objek seperti TensorFlow, Theano, Torch, Caffe, Chainer, dan Computational Network Toolkit. Dari semua sistem, hanya implementasi TensorFlow dapat mendistribusikan komputasi ke dalam beberapa mesin sedangkan sistem yang lain hanya memetakan komputas ke dalam satu mesin (Abadi, et al., 2016).

TensorFlow ditulis dalam bahasa pemrograman C++ yang dapat menyederhanakan proses *deployment* dari model yang sudah dilatih ke dapat berbagai pengaturan produksi, termasuk perangkat seluler. Sistem ini lahir berdasarkan lebih dari seratus proyek dan penelitian dari produk dan layanan Google. Sejak tahun 2015, TensorFlow sudah bersifat sumber terbuka atau *open-source* sehingga bisa digunakan oleh berbagai peneliti di seluruh dunia. Pada tahun 2019, Google meluncurkan TensorFlow 2 di mana terdapat berbagai pembaharuan dalam kerangka kerja antara lain dihilangkannya berbagai *Application Programming Interface* (API) yang berulang dan berbagai API dibuat lebih konsisten. Pembaruan ini memberikan peluang untuk mengevaluasi manfaat peningkatan ke TensorFlow 2 dalam industri retail. Terlebih lagi, pada Mei 2017, Google mengumumkan TensorFlow Lite, yang dirancang khusus untuk perangkat berdaya terbatas seperti perangkat seluler, IoT, dan mikrokontroler. Beberapa fitur utama dari TensorFlow Lite adalah bobot yang kecil, waktu inferensi yang singkat, dan keamanan yang lebih baik.

Dalam kasus deteksi objek untuk perangkat seluler, ada beberapa arsitektur deteksi objek yang sudah terbukti memiliki performa yang optimal, seperti Single Shot MultiBox Detector (SSD) (Liu, et al., 2016). Arsitektur ini memiliki keunggulan dalam hal kecepatan inferensi tanpa mengorbankan akurasi, terutama jika dikombinasikan dengan backbone seperti MobileNet (Howard, et al., 2017). MobileNet dirancang untuk efisiensi dengan menggunakan teknik depthwise separable convolutions, yang membuatnya ideal untuk aplikasi real-time di perangkat seluler. Oleh karena itu, arsitektur SSD MobileNet menjadi pilihan yang populer untuk deteksi objek pada perangkat dengan sumber daya terbatas.

Penulis sebelumnya telah mengimplementasikan deteksi objek untuk produk retail menggunakan arsitektur SSD MobileNet pada TensorFlow versi 1. Namun, penggunaan TensorFlow 1 sudah mulai banyak ditinggalkan oleh komunitas karena hadirnya versi terbaru dari TensorFlow, yakni TensorFlow 2, yang dirilis tahun 2019. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan deteksi objek pada produk retail menggunakan arsitektur SSD MobileNet pada TensorFlow 1 dan TensorFlow 2, serta mengevaluasi apakah peningkatan dari TensorFlow 1 ke TensorFlow 2 dapat meningkatkan akurasi, efisiensi, dan kesesuaian model deteksi objek untuk aplikasi di perangkat dengan sumber daya terbatas.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, dibutuhkan implementasi teknologi deteksi objek pada sistem ketersediaan barang untuk produk retail. Seberapa baik performa model deteksi objek SSD MobileNet dengan menggunakan TensorFlow 2 dibandingkan dengan TensorFlow 1?

1.3 Batasan Masalah

Untuk menjaga fokus dari penelitian ini agar tidak terlalu luas dan, dibutuhkan batasan-batasan masalah, yaitu:

a. Kerangka kerja yang digunakan pada penelitian ini adalah TensorFlow 1 dan 2.
b. Produk yang dijadikan objek deteksi hanya produk retail minuman *Ready-to-Drink* (RTD).
c. Jumlah kelas yang digunakan dalam penelitian ini adalah 100 kelas.
d. Matriks evaluasi yang digunakan adalah *Mean Average Precision* (mAP).
e. Arsitektur yang digunakan adalah SSD MobileNetV3 Large untuk TensorFlow 1 dan SSD MobileNetV2 FPNLite untuk TensorFlow 2.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

a. Mengimplementasikan teknologi deteksi objek dengan SSD MobileNet pada produk retail menggunakan TensorFlow 1 dan 2.
b. Membandingkan performa model yang sudah dilatih antara TensorFlow 1 dan 2.

1.5 Manfaat Penelitian

Beberapa manfaat yang diharapkan dari adanya penelitian ini adalah:

a. Meningkatkan ketersediaan barang retail pada rak.
b. Mencegah terjadinya kekosongan barang.
c. Mengurangi biaya operasional yang dibutuhkan untuk memeriksa ketersediaan barang secara manual.
d. Meningkatkan waktu yang dibutuhkan untuk memeriksa ketersediaan barang.
e. Memberikan insight mengenai penggunaan model SSD MobileNet dengan framework TensorFlow 1 dan 2.
f. Menjadi referensi dari penelitian serupa di masa yang akan datang.

1.6 Metodologi Penelitian

Metodologi dari penelitian ini terdiri dari beberapa tahap yakni:

a. Pengumpulan Data. Pada tahap ini, gambar dikumpulkan guna dijadikan data untuk pelatihan.
b. Pelabelan Data. Pada tahap ini, gambar yang dikumpulkan diberikan anotasi sehingga objek pada gambar dapat dikenali oleh model.
c. Pemrosesan Data. Pada tahap ini, data yang sudah diberi anotasi kemudian dilakukan splitting, resizing, dan pengubahan format ke TFRecord.
d. Pelatihan Data. Pada tahap ini, data yang sudah diberi label kemudian dilakukan pelatihan berdasarkan *hyperparameter* yang sudah ditentukan.
e. Integrasi. Model yang sudah dilatih kemudian diintegrasikan terhadap perangkat bergerak menggunakan format model TFLite.
f. Evaluasi. Model kemudian dievaluasi terhadap data gambar yang tidak termasuk ke dalam data pelatihan, baik itu model asli maupun model TFLite.

1.7 Sistematika Penulisan

Sistematika penulisan dari penelitian ini terdiri dari lima bab, yakni:

BAB I PENDAHULUAN

Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan struktur laporan. Gambaran umum dari penelitian dijelaskan pada bab ini.

BAB II LANDASAN TEORI

Bab ini menjelaskan berbagai teori yang berkaitan dengan penelitian. Sumber teori berasal dari berbagai penelitian yang sudah pernah dilakukan sebelumnya.

BAB III METODOLOGI PENELITIAN

Bab ini berisi tentang metode dalam pengumpulan data, pelabelan data, pemrosesan data, pelatihan data, integrasi model, dan evaluasi model.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil evaluasi yang diperoleh dari penelitian serta pembahasan dari hasil evaluasi tersebut.

BAB IV PENUTUP

Bab ini berisi kesimpulan dari penelitian ini dan saran yang diharapkan bermanfaat terhadap penelitian yang akan dilakukan pada masa mendatang.

BAB II
LANDASAN TEORI

2.1 On-Shelf Availability

On-Shelf Availability (OSA) adalah ukuran jumlah produk yang tersedia dalam kondisi siap jual kepada pelanggan, di tempat yang diharapkannya, dan pada waktu yang diinginkannya untuk membeli produk tersebut. Menjaga Ketersediaan di Rak atau mengurangi Kekosongan Rak (OOS) membantu meningkatkan manajemen inventaris di toko. Kekosongan Rak (OOS), sebagai lawan dari OSA, terjadi ketika seorang konsumen di toko ritel tiba di rak dan produk spesifik yang mereka cari tidak tersedia. Beberapa skenario yang terjadi jika rak kosong dapat dilihat pada Gambar 2.1.

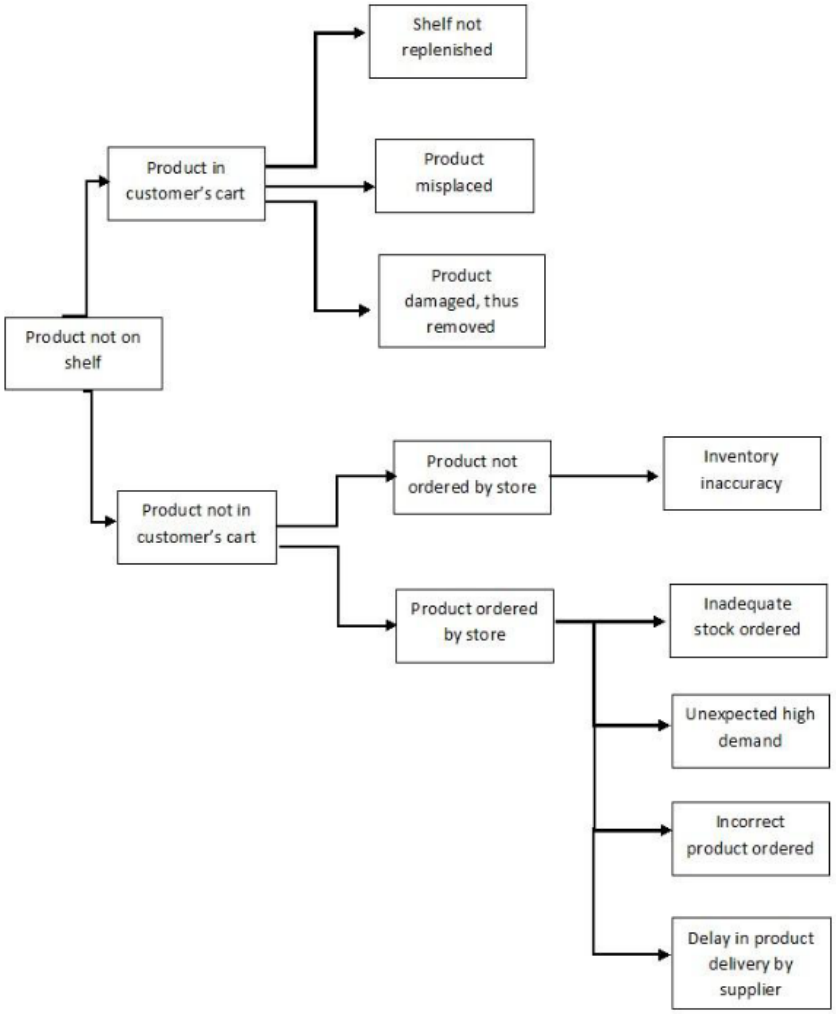
Ada beberapa alasan terjadinya Kekosongan Rak (OOS). Berkembangnya jumlah item retail secara otomatis mengurangi kapasitas penyimpanan per item di rak atau di gudang. Pengurangan ini bersamaan dengan penurunan area gudang oleh banyak pelaku retail yang ingin mendapatkan ruang penjualan tambahan. Penyebab utama lainnya untuk Kekosongan Rak adalah inventaris hantu (*phantom inventory*). Inventaris hantu terjadi ketika sistem manajemen inventaris menunjukkan bahwa suatu produk tertentu tersedia bahkan ketika produk tersebut sebenarnya tidak ada. Hal ini dapat disebabkan jika produk rusak atau dicuri dari rak, jika produk ditarik kembali, atau jika produk masih berada dalam keranjang belanja pelanggan (Moorthy, Behera, & SauravVerma, 2015).

2.2 Computer Vision

Computer vision adalah cabang dari kecerdasan buatan yang bertujuan untuk memungkinkan komputer mengekstrak informasi yang berguna dari gambar dan video untuk mengotomatisasi pemahaman visual. Konsep ini mencakup proses pengambilan data visual, pemrosesan, dan interpretasi untuk mengidentifikasi objek, mendeteksi pergerakan, hingga memahami adegan dalam gambar atau video (Szeliski, 2010).

Computer vision telah digunakan secara luas di berbagai bidang, seperti autonomous driving, pengenalan wajah, analisis citra medis, dan pengawasan. Prosesnya mencakup beberapa langkah dasar, antara lain preprocessing, feature extraction, dan akhirnya pengambilan keputusan berdasarkan pola atau fitur yang teridentifikasi dalam gambar (Forsyth & Ponce, 2012). Berbagai algoritma, seperti edge detection, segmentation, dan deep learning,

telah dikembangkan untuk memaksimalkan kemampuan komputer dalam memahami informasi visual.

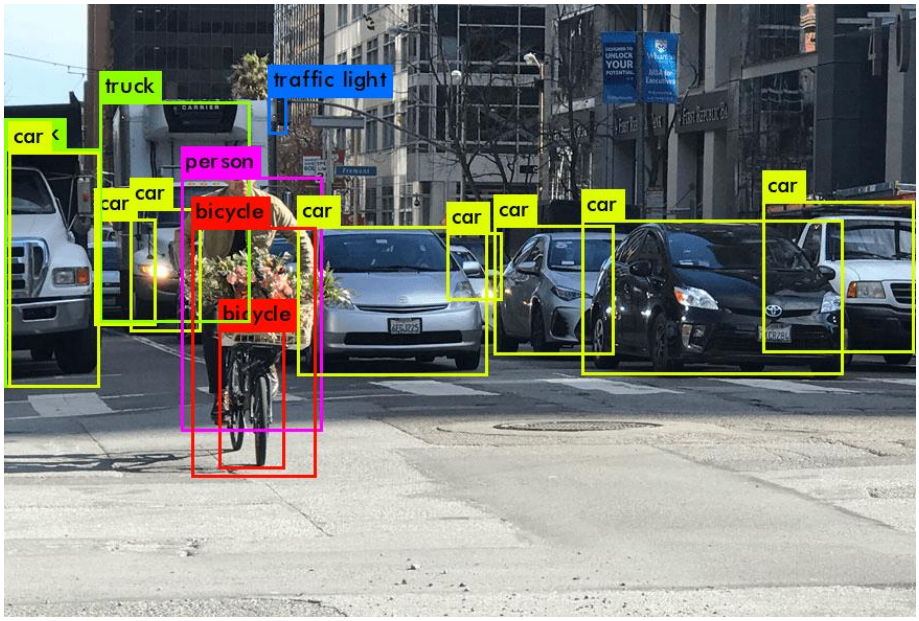


Gambar 2. 1 Skenario yang Terjadi Jika Barang Tidak Ada di Rak.
Sumber: Moorthy, Behera, & SauravVerma (2015)

Dalam beberapa dekade terakhir, perkembangan teknologi deep learning telah membawa kemajuan signifikan dalam computer vision. Metode berbasis deep learning, khususnya Convolutional Neural Networks (CNN), telah menunjukkan kemampuan luar biasa dalam memahami citra visual dan memberikan akurasi tinggi pada berbagai aplikasi klasifikasi gambar dan deteksi objek (LeCun, Bengio, & Hinton, 2015).

2.3 Object Detection

Object detection adalah teknik dalam computer vision yang tidak hanya bertujuan untuk mengklasifikasikan objek dalam suatu gambar, tetapi juga mendeteksi lokasi keberadaan objek tersebut melalui bounding box. Teknik ini penting untuk aplikasi yang memerlukan informasi tentang posisi objek di dalam gambar atau video, seperti sistem pengawasan, kendaraan otonom, dan analisis ritel (Liu, et al., 2016). Contoh implementasi object detection dapat dilihat pada Gambar 2.2.



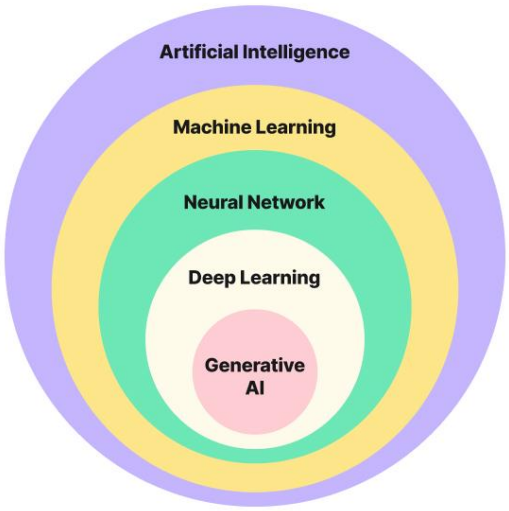
Gambar 2. 2 Contoh Penggunaan Teknologi Object Detection.
Sumber: visionplatform.ai

Object detection menggabungkan dua proses, yaitu klasifikasi dan lokalisasi objek. Dalam deep learning, beberapa arsitektur populer yang digunakan untuk object detection antara lain adalah Faster R-CNN, YOLO (You Only Look Once), dan SSD (Single Shot Detector). Setiap arsitektur memiliki karakteristik unik, seperti YOLO yang terkenal dengan kecepatannya (Redmon, Divvala, Girshick, & Farhadi, 2016), dan Faster R-CNN yang unggul dalam akurasi namun membutuhkan waktu inferensi yang lebih lama (Ren, He, Girshick, & Sun, 2017).

Seiring perkembangan teknologi deep learning, model object detection semakin efisien dan akurat dengan bantuan teknik seperti Feature Pyramid Networks (FPN) dan penggunaan backbone jaringan konvolusi yang lebih dalam, seperti ResNet dan MobileNet. Teknik-teknik ini memungkinkan deteksi objek pada berbagai skala dan memperbaiki performa model dalam mendeteksi objek kecil di dalam gambar (Lin, et al., 2017).

2.4 Deep Learning

Deep Learning adalah subbidang Machine Learning yang memanfaatkan jaringan saraf buatan (*Artificial Neural Network* atau ANN) berskala besar dan berlapis-lapis untuk memproses data melalui representasi kontinu dalam bentuk bilangan real. Machine Learning sendiri merupakan bagian dari Artificial Intelligence seperti yang terlihat pada Gambar 2.3. Jaringan Deep Learning ini meniru struktur hierarkis neuron dalam otak manusia, di mana setiap lapisan menangkap pola fitur yang semakin kompleks dari data yang diproses (LeCun, Bengio, & Hinton, 2015).



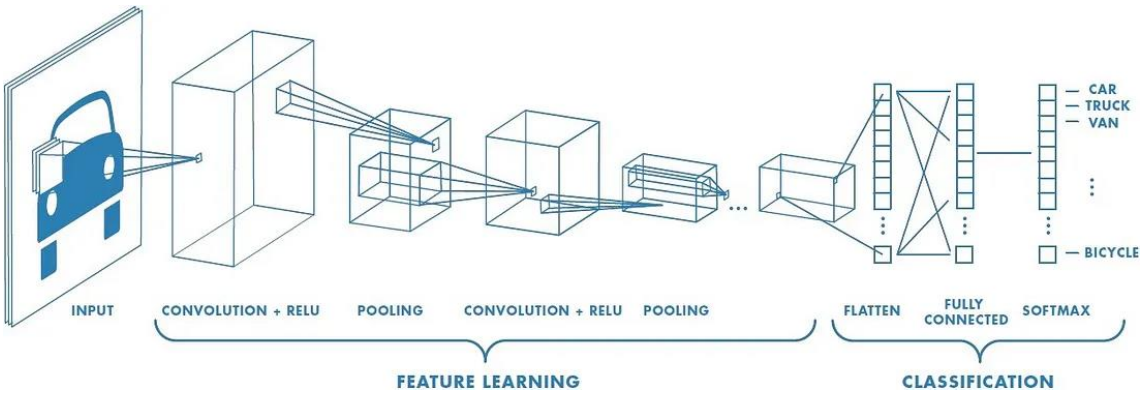
Gambar 2. 3 Deep Learning Merupakan Bagian dari Machine Learning.
Sumber: dicoding.com

Deep Learning telah menjadi pendekatan yang paling berhasil dalam Machine Learning saat ini karena kemampuannya untuk melakukan generalisasi dengan baik, bahkan pada dataset yang besar dan kompleks, serta skalabilitasnya yang tinggi dengan peningkatan daya komputasi. Aplikasinya mencakup pengenalan suara, pengenalan wajah, analisis teks, serta deteksi objek dalam gambar, menjadikannya salah satu teknologi utama di berbagai industri. Dalam konteks deteksi objek, model Deep Learning dapat menangkap pola fitur pada gambar dengan sangat mendalam sehingga dapat menghasilkan prediksi yang lebih akurat.

2.5 Convolutional Neural Network

Convolutional Neural Networks (CNN) serupa dengan *Artificial Neural Network* (ANN) tradisional karena terdiri dari neuron-neuron yang mengoptimalkan diri melalui pembelajaran. Setiap neuron masih akan menerima input dan melakukan operasi dasar dari banyak ANN. Mulai dari vektor gambar mentah input hingga output akhir skor kelas, seluruh jaringan masih akan menyatakan fungsi skor bobotnya. Lapisan terakhir akan berisi *loss function* yang berhubungan dengan kelas yang tersedia, dan semua tips dan trik yang dikembangkan untuk ANN tradisional masih berlaku.

Perbedaan utama antara CNN dan ANN tradisional hanya terletak pada fakta bahwa CNN utamanya digunakan dalam bidang pengenalan pola dalam gambar. Ini memungkinkan kita untuk menyandikan fitur khusus gambar ke dalam arsitektur, membuat jaringan lebih cocok untuk tugas-tugas yang berfokus pada gambar - sambil lebih mengurangi parameter yang diperlukan untuk menyiapkan model (O'Shea & Nash, 2015). Contoh gambaran umum arsitektur dari Convolutional Neural Network dapat dilihat pada Gambar 2.4.



Gambar 2. 4 Gambaran Umum Arsitektur Convolutional Neural Network.

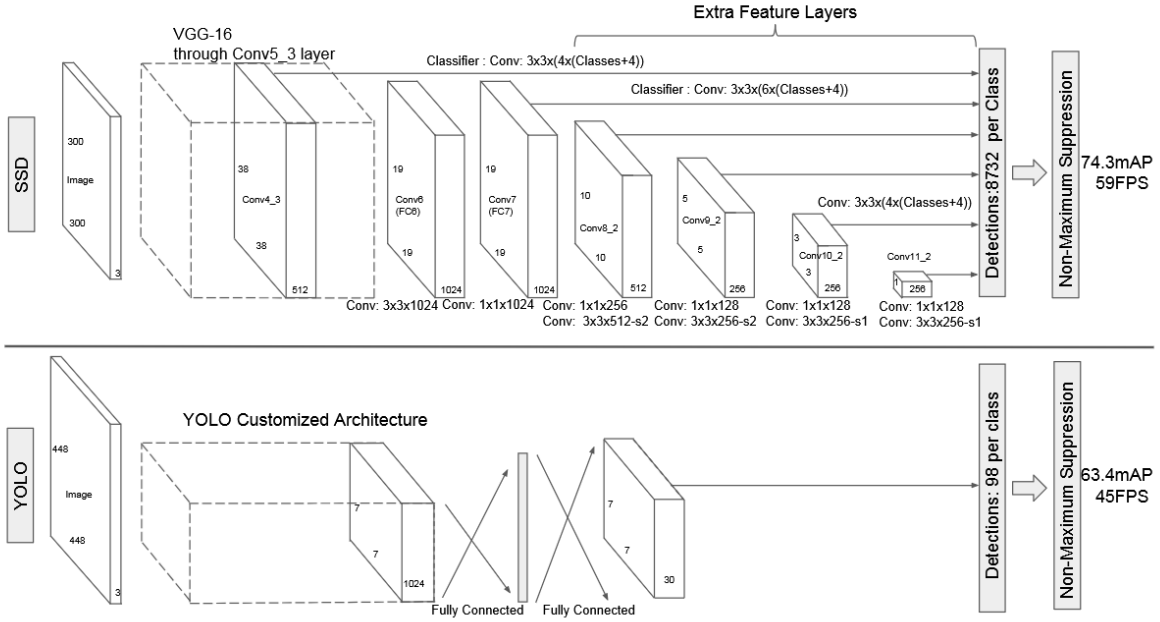
Sumber: towardsdatascience.com

2.5.1 Single Shot Detector

Single Shot Detector (SSD) adalah arsitektur deep learning yang populer untuk deteksi objek secara real-time. Arsitektur ini diperkenalkan oleh Liu et al. pada tahun 2016 dan dirancang untuk menghasilkan prediksi bounding box dan kelas objek dalam satu proses langsung, tanpa memerlukan tahapan pemrosesan berulang seperti pada metode deteksi objek sebelumnya, misalnya Faster R-CNN. SSD melakukan deteksi objek dengan mendeteksi lokasi objek dan menentukan kelas objek sekaligus, yang membuatnya efisien dan cocok untuk

aplikasi real-time pada perangkat berdaya rendah. Arsitektur dari SSD dapat dilihat pada Gambar 2.5.

SSD menggunakan pendekatan multiple feature maps dari beberapa lapisan dalam jaringan untuk menangkap informasi objek pada berbagai skala. Hal ini memungkinkan SSD untuk mendeteksi objek dalam berbagai ukuran dan jarak pandang. Arsitektur SSD menggunakan convolutional layers untuk membangkitkan set prediksi bounding box, yang memungkinkan deteksi lebih cepat karena operasi ini dapat diparalelkan (Liu, et al., 2016).



Gambar 2. 5 Perbedaan Arsitektur SSD dengan YOLO.
Sumber: Liu, et al. (2016)

Dibandingkan dengan arsitektur lain seperti Faster R-CNN, SSD memiliki kecepatan inferensi yang lebih tinggi, meskipun dengan sedikit kompromi pada akurasi. Oleh karena itu, SSD banyak digunakan dalam berbagai aplikasi mobile dan embedded, seperti pada optimasi ketersediaan produk di rak dalam industri retail yang memerlukan deteksi objek secara cepat dan efisien (Huang, et al., 2017).

2.5.2 MobileNet

MobileNet adalah keluarga arsitektur CNN yang dirancang untuk perangkat dengan daya komputasi terbatas, seperti ponsel dan perangkat IoT. Arsitektur ini dikembangkan oleh Google dengan tujuan untuk menyediakan solusi efisien bagi aplikasi berbasis visi komputer di

perangkat mobile, tanpa mengorbankan akurasi secara signifikan. MobileNet memiliki beberapa versi, dengan setiap versi menawarkan peningkatan efisiensi dan akurasi.

MobileNetV1 memperkenalkan konsep depthwise separable convolutions, yang bertujuan untuk mengurangi jumlah parameter dan operasi komputasi dalam model. Dengan menggunakan convolusi depthwise, MobileNetV1 secara drastis mengurangi kebutuhan komputasi dibandingkan dengan konvolusi konvensional. Arsitektur dari MobileNetV1 dapat dilihat pada Gambar 2.6. Meskipun ringan, MobileNetV1 tetap mampu menghasilkan performa yang cukup baik dalam tugas-tugas klasifikasi dan deteksi objek, terutama di perangkat mobile (Howard, et al., 2017).

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

Gambar 2. 6 Body Arsitektur MobileNet.

Sumber: Howard, et al. (2017)

MobileNetV2 membawa peningkatan dengan memperkenalkan blok inverted residuals dan lapisan linear bottleneck. Struktur ini mempertahankan efisiensi komputasi, sekaligus meningkatkan kemampuan jaringan dalam mengekstraksi fitur penting. Strutktur dari MobileNetV2 dapat dilihat pada Gambar 2.7. MobileNetV2 dirancang agar lebih optimal untuk transfer learning dan lebih baik dalam menghindari masalah information loss, sehingga

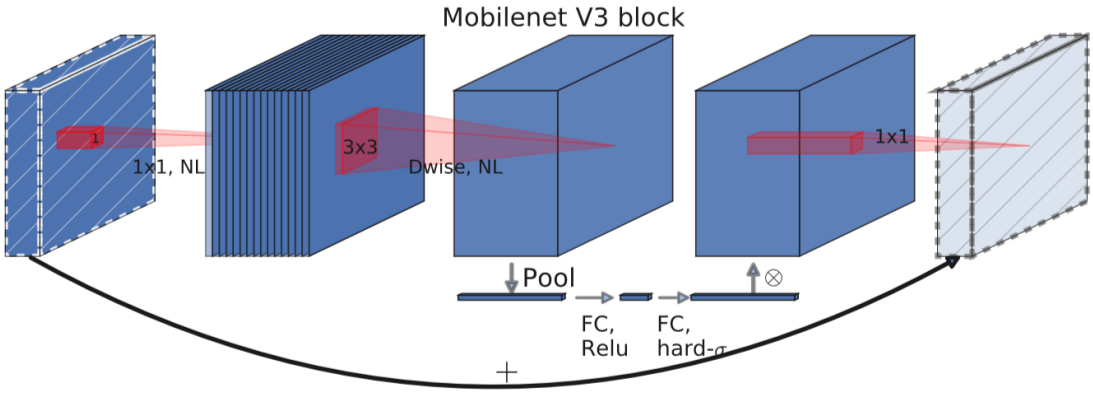
memberikan akurasi yang lebih tinggi dibandingkan versi sebelumnya pada berbagai tugas, termasuk klasifikasi dan deteksi objek (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018).

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Gambar 2. 7 Arsitektur dari MobileNetV2.

Sumber: Sandler, et al. (2018)

MobileNet V3 merupakan penyempurnaan dari versi sebelumnya dengan menggabungkan teknik arsitektur modern seperti platform-aware network architecture search dan NetAdapt. MobileNet V3 hadir dalam dua versi, yaitu MobileNet V3-Large dan MobileNet V3-Small, yang masing-masing dioptimalkan untuk kebutuhan berbeda, dengan fokus pada keseimbangan antara akurasi dan efisiensi. MobileNet V3 juga memanfaatkan squeeze-and-excitation modules untuk meningkatkan sensitivitas model terhadap fitur-fitur penting pada gambar (Howard, et al., 2019). Blok pada MobileNet V3 dapat dilihat pada Gambar 2.8.



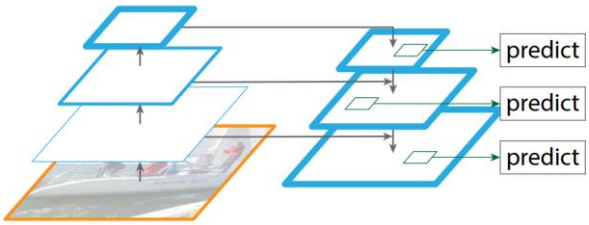
Gambar 2. 8 Blok dari MobileNetV3.

Sumber: Howard, et al. (2019)

Dalam penelitian ini, penulis menggunakan MobileNetV2 untuk TensorFlow 2 dan MobileNetV3 untuk TensorFlow 1. Keduanya dipilih karena masing-masing merupakan versi arsitektur terbaru yang dirilis di dalam TensorFlow Object Detection API berdasarkan versi TensorFlownya (Huang, et al., 2017). Dengan memilih arsitektur dengan versi terbaru, diharapkan dapat menjaga proses pemeliharaan model untuk jangka waktu yang lebih lama.

2.5.3 Feature Pyramid Network

Feature Pyramid Network (FPN) adalah arsitektur yang dikembangkan untuk meningkatkan performa model dalam mendeteksi objek pada berbagai skala. FPN menggunakan piramida fitur untuk menggabungkan informasi dari level-level yang berbeda dalam jaringan konvolusi, yang memungkinkan model menangani variasi ukuran objek secara efektif. Ilustrasi cara kerja FPN dapat dilihat pada Gambar 2.9. Teknik ini sangat bermanfaat dalam mendeteksi objek kecil yang sulit dikenali pada layer konvolusi dengan resolusi rendah (Lin, et al., 2017).



Gambar 2. 9 Ilustrasi Cara Kerja Feature Pyramid Network.
Sumber: Lin, et al. (2017)

FPN memanfaatkan dua aliran pemrosesan utama: jalur bawah (bottom-up pathway) dan jalur atas (top-down pathway). Jalur bawah berfungsi untuk mengekstrak fitur dasar dari gambar, sedangkan jalur atas memanfaatkan fitur ini dan menerapkannya dalam skala yang lebih tinggi dengan menggabungkan informasi kontekstual dari layer-layer sebelumnya. FPN juga menggunakan lateral connections untuk menggabungkan fitur dari berbagai resolusi, menghasilkan representasi yang lebih kaya untuk setiap skala objek. Arsitektur ini telah terbukti efektif dalam meningkatkan akurasi deteksi objek, terutama dalam kombinasi dengan model deteksi seperti Faster R-CNN dan Mask R-CNN.

FPNLite adalah versi yang lebih ringan dari FPN, dirancang untuk aplikasi yang membutuhkan efisiensi tinggi, seperti pada perangkat mobile. FPNLite menggunakan prinsip

dasar FPN, tetapi dengan beberapa penyesuaian untuk mengurangi jumlah parameter dan komputasi. FPNLite sering digunakan dengan arsitektur yang lebih ringan, seperti SSD MobileNet, untuk memastikan inferensi cepat pada perangkat dengan keterbatasan daya komputasi. Dengan FPNLite, model dapat tetap mempertahankan akurasi yang baik sambil meningkatkan efisiensi, yang menjadikannya pilihan tepat untuk aplikasi real-time (Howard et al., 2019).

2.6 TensorFlow

TensorFlow adalah pustaka open-source yang dikembangkan oleh Google Brain untuk kebutuhan machine learning dan deep learning. TensorFlow mendukung pengembangan model yang dapat dijalankan di berbagai perangkat, mulai dari desktop hingga perangkat mobile. Sejak diluncurkan pertama kali pada 2015, TensorFlow telah menjadi salah satu pustaka populer dalam komunitas pengembang dan peneliti deep learning. Berbagai perusahaan besar menggunakan TensorFlow untuk menjalankan proses bisnis mereka seperti yang telihat pada Gambar 2.10.

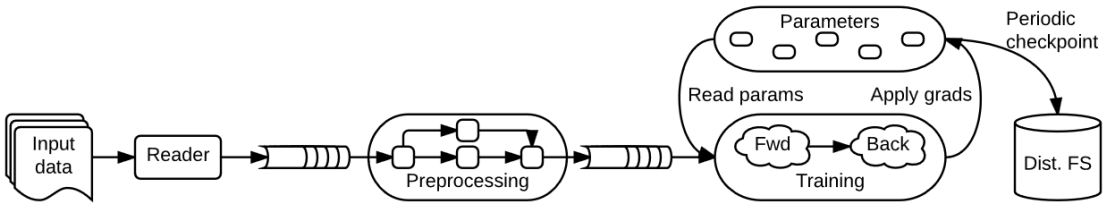
Gambar 2. 10 Beberapa Perusahaan yang Menggunakan TensorFlow.
Sumber: towardsdatascience.com

TensorFlow beroperasi dalam skala besar di lingkungan pengembangan yang heterogen. TensorFlow menggunakan grafik dataflow untuk merepresentasikan komputasi, status, dan

operasi yang mengubah status tersebut. Sistem ini memetakan node-node dari grafik dataflow ke banyak mesin dalam sebuah klaster, dan dalam satu mesin melintasi beberapa perangkat komputasi, termasuk CPU, GPU, dan TPU (*Tensor Processing Units*). TensorFlow memberikan fleksibilitas kepada pengembang untuk bereksperimen dengan berbagai optimisasi dan algoritma pelatihan baru (Abadi, et al., 2016).

2.6.1 TensorFlow 1

TensorFlow 1 menawarkan banyak fitur untuk pengembangan dan pelatihan model, tetapi dengan beberapa keterbatasan dalam hal fleksibilitas. TensorFlow 1 menggunakan graph execution, di mana model harus didefinisikan dan dibangun dalam bentuk computational graph yang statis sebelum dieksekusi. Pendekatan ini memungkinkan optimasi performa untuk skala besar, tetapi membuat proses debugging dan pengembangan model menjadi lebih kompleks dan kurang intuitif. Selain itu, TensorFlow 1 mendukung fitur-fitur seperti sync_replicas untuk pengaturan replikasi dalam training dengan GPU, yang berguna dalam meningkatkan efisiensi training pada lingkungan multi-GPU. Untuk contoh dari grafik pelatihan pada TensorFlow dapat dilihat pada Gambar 2.11.



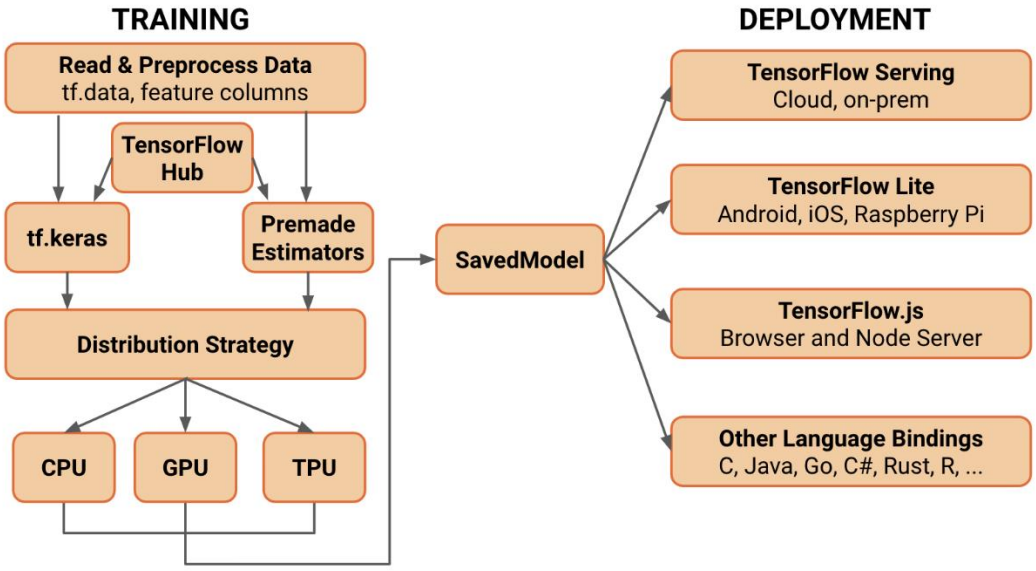
Gambar 2. 11 Grafik Pelatihan Data pada TensorFlow.
Sumber: Abadi, et al. (2016)

2.6.2 TensorFlow 2

TensorFlow 2 dirilis dengan banyak perbaikan yang membuat proses pengembangan model menjadi lebih user-friendly dan efisien. Salah satu perubahan utama adalah diperkenalkannya eager execution, di mana operasi dapat dijalankan secara langsung tanpa harus membangun computational graph terlebih dahulu. Ini memungkinkan proses debugging yang lebih mudah dan pengembangan model yang lebih intuitif. TensorFlow 2 juga mendukung integrasi penuh dengan Keras sebagai API tingkat tinggi, memudahkan para pengembang

untuk membangun, melatih, dan mengevaluasi model dengan lebih sedikit baris kode (TensorFlow Dev Team, 2019).

Selain itu, TensorFlow 2 memberikan peningkatan pada performa model dalam deployment di perangkat mobile. Meskipun TensorFlow Lite (TFLite) sudah tersedia sejak TensorFlow 1, integrasinya semakin disempurnakan di TensorFlow 2, sehingga lebih mudah digunakan untuk konversi dan optimalisasi model pada perangkat mobile atau embedded systems. Perubahan-perubahan ini membuat TensorFlow 2 lebih ideal untuk pengembangan aplikasi machine learning di berbagai platform, baik untuk penelitian maupun produksi. Diagram konsep dari arsitektur baru di TensorFlow 2 dapat dilihat pada Gambar 2.12.



Gambar 2. 12 Diagram Konsep Arsitektur dari TensorFlow 2.

Sumber: blog.tensorflow.org

2.7 Metrik Evaluasi

Dalam pengembangan model deteksi objek, metrik evaluasi memiliki peran yang sangat penting untuk mengukur performa model terhadap dataset yang digunakan. Metrik evaluasi digunakan untuk menentukan sejauh mana model mampu mendeteksi objek secara akurat, baik dari segi klasifikasi maupun dari segi lokasi bounding box. Evaluasi ini membantu peneliti memahami kekuatan dan kelemahan model yang dikembangkan serta memberikan gambaran untuk peningkatan performa di masa mendatang. Pada penelitian ini, beberapa metrik evaluasi utama yang digunakan meliputi Mean Average Precision (mAP), Average Precision (AP),

Precision, Recall, Intersection over Union (IoU), serta kerugian seperti Classification Loss, Localization Loss, dan Total Loss.

2.7.1 Mean Average Precision (mAP)

Mean Average Precision (mAP) adalah metrik utama yang digunakan untuk mengevaluasi model deteksi objek. Metrik ini merupakan rata-rata dari nilai Average Precision (AP) untuk semua kelas yang ada dalam dataset. mAP sering kali dihitung pada berbagai nilai threshold Intersection over Union (IoU), seperti mAP@0.5 (IoU ≥ 0.5) atau mAP@[0.5:0.95] (rata-rata mAP pada IoU dari 0.5 hingga 0.95 dengan interval 0.05). Pentingnya mAP terletak pada kemampuannya untuk memberikan gambaran keseluruhan tentang akurasi model dalam mendeteksi dan mengklasifikasikan objek dari berbagai kelas, menjadikannya standar evaluasi dalam banyak penelitian deteksi objek. Rumus untuk menghitup mAP dapat dilihat pada persamaan 2.1.

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \qquad (2.1)$$

Keterangan:

- mAP : Mean Average Precision, rata-rata nilai AP untuk semua kelas.
- N : Jumlah kelas atau kategori dalam dataset.
- APi : Average Precision untuk kelas ke-i.

2.7.2 Average Precision (AP)

Average Precision (AP) adalah metrik yang menghitung rata-rata nilai Precision pada berbagai nilai Recall. Untuk setiap kelas, AP mengukur kemampuan model dalam mendeteksi objek dengan tingkat keakuratan yang bervariasi. AP dihitung dengan menggunakan kurva Precision-Recall, di mana area di bawah kurva tersebut (Area Under Curve, AUC) merepresentasikan nilai AP. AP untuk setiap kelas kemudian dirata-ratakan untuk mendapatkan mAP. Rumus dari AP dapat dilihat pada persamaan 2.2.

$$\mathrm{AP} = \frac{1}{M}\sum_{j=1}^{M} \mathrm{Precision}(j) \cdot \Delta\mathrm{Recall}(j) \quad (2.2)$$

Keterangan:

- AP : Average Precision, rata-rata nilai precision pada berbagai nilai recall.
- M : Jumlah titik (threshold) evaluasi dalam kurva Precision-Recall.
- Precision(j) : Nilai precision pada titik evaluasi ke-j.
- ΔRecall(j) : Perubahan nilai recall antara titik evaluasi j dan $j-1$.

2.7.3 Precision

Precision adalah proporsi dari prediksi yang benar-benar positif dibandingkan dengan semua prediksi positif yang dihasilkan oleh model. Metrik ini penting dalam aplikasi di mana kesalahan prediksi positif (False Positive) harus diminimalkan. Precision yang tinggi menunjukkan bahwa sebagian besar prediksi positif model adalah benar. Rumus dari precision dapat dilihat pada persamaan 2.3.

$$\mathrm{Precision} = \frac{\mathrm{True\ Positives\ (TP)}}{\mathrm{True\ Positives\ (TP)} + \mathrm{False\ Positives\ (FP)}} \quad (2.3)$$

Keterangan:

- Precision : Proporsi Prediksi positif yang benar dari semua prediksi positif.
- TP (True Positives) : Jumlah sampel yang diprediksi positif dan benar.
- FP (False Positives) : Jumlah sampel yang diprediksi positif tetapi salah.

2.7.4 Recall

Recall adalah proporsi dari objek yang berhasil dideteksi oleh model dibandingkan dengan seluruh objek yang sebenarnya ada dalam dataset. Recall berguna untuk memahami sejauh mana model mampu menemukan semua objek target dalam dataset. Recall yang tinggi menunjukkan bahwa model mampu mendeteksi sebagian besar objek dalam dataset. Rumus dari recall dapat dilihat pada persamaan 2.4.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \quad (2.4)$$

Keterangan:

- Recall : Proporsi sampel positif yang berhasil dideteksi dengan benar.
- TP (True Positives) : Jumlah sampel yang diprediksi positif dan benar.
- FN (False Negatives) : Jumlah sampel yang seharusnya positif tetapi diprediksi negatif.

2.7.5 Intersection over Union (IoU)

Intersection over Union (IoU) adalah metrik yang digunakan untuk mengukur sejauh mana bounding box prediksi sesuai dengan bounding box ground truth. IoU dihitung sebagai rasio antara area overlap dan area union dari bounding box prediksi dan ground truth. IoU biasanya digunakan sebagai threshold untuk menentukan apakah sebuah prediksi dianggap benar (True Positive) atau salah (False Positive). IoU dapat dihitung dengan rumus yang terdapat pada persamaan 2.5.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \quad (2.5)$$

Keterangan:

- IoU : Metrik overlap antara bounding box prediksi (A) dan ground truth (B).
- $|A \cap B|$: Luas area yang merupakan overlap antara prediksi dan ground truth.
- $|A \cup B|$: Luas gabungan area bounding box prediksi dan ground truth.

2.7.6 Classsification Loss

Classification Loss adalah kerugian yang mengukur kesalahan model dalam mengklasifikasikan objek ke kelas yang benar. Kerugian ini biasanya dihitung menggunakan fungsi seperti Cross-Entropy Loss. Kerugian ini penting untuk memastikan bahwa model dapat mengklasifikasikan objek dengan tingkat akurasi yang tinggi.

2.7.7 Localization Loss

Localization Loss adalah kerugian yang mengukur sejauh mana bounding box prediksi berbeda dari bounding box ground truth. Kerugian ini biasanya dihitung menggunakan fungsi

Smooth L1 Loss atau L2 Loss. Localization Loss penting untuk memastikan bahwa bounding box prediksi model mendekati lokasi sebenarnya dari objek yang terdeteksi.

2.7.8 Total Loss

Total Loss adalah kombinasi dari Classification Loss dan Localization Loss yang digunakan untuk melatih model. Total Loss memberikan gambaran keseluruhan tentang seberapa baik model mampu mendeteksi dan mengklasifikasikan objek selama proses pelatihan. Semakin kecil nilai Total Loss, semakin baik performa model dalam menyelesaikan tugas deteksi objek.

2.8 Studi Literatur

Studi literatur adalah bentuk penelitian terhadap berbagai publikasi yang memiliki kesamaan topik dengan penelitian yang akan dilakukan. Dalam deteksi objek untuk produk retail, ada beberapa penelitian yang sudah dilakukan baik itu dari dalam negeri maupun luar negeri. Adapun penelitian lain yang memanfaatkan arsitektur SSD MobileNet juga akan dicantumkan pada studi literatur ini.

Sebuat penelitian di Universitas Telkom pada tahun 2021 pernah mengangkat topik pengenalan objek untuk produk retail. Penelitian ini menggunakan arsitektur yang digunakan adalah YOLOv3 dan mendapatkan hasil yang cukup baik pada objek tunggal dengan akurasi mencapai 90%. Sedangkan untuk gambar dengan kondisi objek yang banyak, akurasinya mencapai 80% (Ardiansyah, Muttaqin, Prasetio, & Novitasari, 2021).

Penelitian lain di Universitas Dokuz Eylül, Turki pada tahun 2021 mengangkat topik pelabelan gambar secara otomatis pada produk retail. Dalam penelitian ini, arsitektur yang digunakan terdiri dari RetinaNet, YOLOv3, dan YOLOv4. Hasil terbaik diperoleh oleh YOLOv4 dengan skor mAP sebesar 0.9187 (Yilmazer & Birant, 2021).

Walaupun YOLO sudah membuktikan keunggulannya dalam hal mendeteksi objek, namun mAP yang dihasilkan YOLO untuk mendeteksi objek belum sebaik SSD, khususnya untuk mendeteksi objek dengan ukuran kecil. Hal ini dibuktikan dengan penelitian yang dilakukan oleh Liu et al pada tahun 2017. Baik YOLO maupun SSD memiliki kesamaan yakni keduanya menggunakan pendekatan one stage (Sabina, Aneesa, & Haseena, 2022).

Penggunaan SSD MobileNet dengan FPN juga sudah dilakukan di beberapa skenario penelitian, salah satunya deteksi sampah yang dilakukan oleh Meng, et al. pada tahun 2021. Penelitian tersebut menggunakan dataset sampah yang terdiri dari 1000 gambar dan terbagi

menjadi empat kategori. Metode yang digunakan di penelitian ini adalah SSD MobileNet dengan FPN dan tanpa FPN. Hasil penelitian ini menunjukkan bahwa model yang memanfaatkan FPN mampu meningkatkan akurasi, khususnya pada objek kecil. Model yang menggunakan FPN juga mampu melakkukan deteksi dengan inference time yang lebih cepat (Meng, Jiang, Wang, & Wang, 2021).

Ada penelitian pada produk retail dengan menggunakan SSD MobileNet yang dilakukan oleh Gianluca Fighera pada tahun 2019. Pada penelitian tersebut, dilakukan terhadap dataset yang terdiri dari 16 SKU parfum yang diletakkan di rak. Penelitian tersebut memanfaatkan beberapa arsitektur yang tersedia di TensorFlow Object Detection API seperti SSD MobileNetV2, Faster R-CNN ResNet50, dan Faster R-CNN InceptionV2. Hasil dari penelitian tersebut menunjukkan bahwa model yang menggunakan SSD MobileNetV2 mendapatkan waktu inferensi yang paling cepat walaupun paling rendah dari segi akurasi (Fighera, 2019).

Berdasarkan berbagai penelitian sejenis yang sudah ada sebelumnya, terlihat bahwa teknologi deteksi objek dapat diimplementasikan dengan baik pada studi kasus deteksi objek untuk produk retail. Arsitektur YOLO dapat melakukan deteksi dengan cukup baik. Sedangkan SSD dapat melakukan deteksi dengan perangkat bergerak secara lebih efisien dibandingkan YOLO. Rangkuman dari berbagai penelitian tersebut dapat dilihat pada Tabel 2.1.

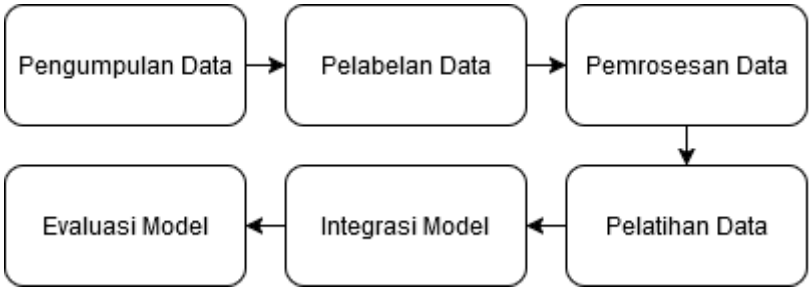
Tabel 2. 1 Daftar Literatur Deteksi Objek Pada Produk Retail dan SSD MobileNet

Penulis	Tahun	Dataset	Arsitektur	Hasil
Ardiansyah, Muttaqin, Prasetio, & Novitasari	2021	Gambar produk yang diambil secara mandiri baik dengan kamera dalam ruangan maupun dari internet	YOLOv3	Akurasi >90% pada objek tunggak dan >80% pada objek jamak
Yilmazer & Birant	2021	WebMarket dataset	RetinaNet, YOLOv3, dan YOLOv4	YOLOv4 mendapatkan mAP sebesar 0.9187
Sabina, Aneesa, & Haseena	2022	Berbagai Skenario yang Berbeda	YOLO, SSD MobileNet	SSD MobileNet lebih cepat dibandingkan YOLO
Meng, Jiang, Wang, & Wang	2021	Dataset Sampah	SSD MobileNet dengan dan tanpa FPN	Penggunaan FPN meningkatkan akurasi dan juga inference time
Fighera	2019	Dataset Parfum	SSD MobileNetV2, Faster R-CNN ResNet50, dan Faster R-CNN InceptionV2	SSD MobileNetV2 mendapat waktu inferensi paling cepat walaupun akurasi paling rendah.

Perbedaan antara penelitian ini dengan penelitian-penelitian sebelumnya adalah fokus utama penelitian ini yang membandingkan performa antara SSD MobileNetV3 Large di TensorFlow 2 dengan SSD MobileNetV2 FPNLite di TensorFlow 1, khsusunya pada studi kasus produk retail. Kedua arsitektur tersebut diambil dari TensorFlow Object Detection API dan merupakan arsitektur SSD MobileNet dengan versi yang paling baru di masing-masing versi TensorFlow-nya. Dengan mengambil fokus komparasi terhadap kedua versi TensorFlow, penelitian ini diharapkan dapat menjadi rujukan untuk penggunaan TensorFlow di masa yang akan datang.

BAB III
METODOLOGI PENELITIAN

Pada Bab ini, metodologi penelitian akan diterapkan berdasarkan rumusan masalah. Terdapat lima tahapan dalam penelitian ini, yakni pengumpulan data, pelabelan data, pemrosesan data, pelatihan data, integrasi model dan evaluasi model. Diagram alir dari tahapan tersebut dapat dilihat pada Gambar 3.1.



Gambar 3. 1 Diagram Alir Metodologi Penelitian

Tahap pertama dari penelitian ini adalah pengumpulan data. Pada tahap ini, data gambar dikumpulkan sebagai bahan untuk dilakukan pelatihan terdahap model deteksi objek. Tahap kedua yakni pelabelan data. Gambar yang sudah dikumpulkan kemudian diberikan anotasi terhadap setiap objek di dalam gambar yang akan dideteksi oleh model. Tahap ketiga yakni pemrosesan data. Pada tahap ini, data yang sudah dianotasi dipisah menjadi data latih dan data uji. Selain itu, pada tahap ini juga dilakukan resizing dan pengubahan format ke TFRecord.

Tahap keempat yakni Pelatihan Data. Pada tahap ini, model mulai dilatih berdasarkan data-data yang sudah diproses pada tahap sebelumnya. Tahap kelima yakni Integrasi Model. Pada tahap ini, model ayng sudah dilatih diubah ke dalam format TFLite untuk ditanam pada perangkat bergerak. Tahap terakhir, yakni Evaluasi Model. Pada tahap ini, baik model asli maupun model TFLite dievaluasi terhadap data uji sehingga performa dari setiap model dapat diukur dan dapat dijadikan referensi untuk penggunaan deteksi objek di masa yang akan datang.

Pada penelitian ini, spesifikasi perangkat yang digunakan untuk tahap pemrosesan data, pelatihan data, dan evaluasi model adalah CPU Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz, RAM sebesar 32 GB, dan GPU NVIDIA Tesla V100 SXM2 32 GB. Sistem operasi yang digunakan adalah Ubuntu 20.04.5 LTS dengan kernel Linux 5.4.0-187-generic. Tahap Integrasi model dilakukan dengan menggunakan perangkat smartphone Samsung A55 5G.

3.1 Pengumpulan Data

Pengumpulan data dilakukan dengan mengambil gambar langsung dari minimarket atau toko-toko yang berisi produk-produk di rak gondola dan showcase cooler. Contoh untuk rak gondola dapat dilihat pada Gambar 3. 2 dan untuk showcase cooler dapat dilihat pada Gambar 3. 3. Penggunaan kedua tempat penyimapan ini mempertimbangkan jenis produk yang akan dideteksi adalah produk *Food and Beverage* (FnB) dengan kategori *Ready-to-Drink* (RTD).

Gambar 3. 2 Contoh Rak Gondola yang dipakai untuk Meletakkan Produk.
Sumber: anddisplay.com

Proses pengambilan gambar dilakukan dengan mengikuti aturan yang ada pada subbab 3.1.1. Aturan ini bertujuan agar gambar memiliki fitur-fitur yang dibutuhkan untuk pelatihan sehingga model mengenali objek-objek yang terdapat di dalam gambar sesuai dengan sifat gambar yang diambil. Gambar-gambar yang dikumpulkan harus memiliki fitur yang dapat dikenali oleh perangkat smartphone karena pada implementasinya, perangkat smartphone-lah yang akan digunakan dalam proses deteksi gambar.

Gambar 3. 3 Contoh Showcase Cooler yang dipakai untuk Meletakkan Produk.
Sumber: duniamasak.com

3.1.1 Aturan Pengumpulan Data

Data gambar yang dikumpulkan memiliki beberapa variasi yang diatur, seperti tingkat kemiringan dimulai dari tegak lurus, miring kanan, miring kiri, miring atas, dan miring bawah. Dengan berbagai Tingkat kemiringan, posisi tegak lurus tetap menjadi prioritas utama agar produk dapat dikenali dengan lebih jelas, khususnya pada planogram dalam produk.

Selain tingkat kemiringan, jarak antara kamera smartphone dengan rak juga perlu disesuaikan, mulai dari foto satu baris rak hingga enam baris rak. Semakin dekat jarak antara kamera dengan rak, maka semakin banyak fitur yang dapat dikenali dari satu objek produk. Dengan demikian, proses pengambilan gambar tetap memprioritaskan jarak antara kamera dengan produk yang tidak terlalu jauh.

Tingkat pencahayaan juga menjadi salah satu aturan yang perlu dipertimbangkan dalam proses pengumpulan data. Pada umumnya, tingkat pencahayaan yang diambil merupakan pencahayaan yang sama dengan cahaya ruangan baik itu di minimarket maupun toko-toko lainnya. Dengan mempertimbangkan Tingkat kemiringan, jarak, dan pencahayaan yang bervariasi dan relevan, data yang dikumpulkan dapat dikenali oleh model sesuai dengan kondisi yang ada di lapangan.

3.1.2 Kelas Dataset

Kelas dataset merupakan representasi dari kategori atau label yang digunakan untuk membedakan antara satu objek dengan objek yang lain di dalam gambar. Daftar dari kelas yang digunakan dalam penelitian ini dapat dilihat pada Tabel 3.1. Kelas dataset yang dikumpulkan berjumlah 100 kelas dan masing-masing kelas memiliki id tersendiri.

Tabel 3. 1 Daftar Kelas dalam Dataset

ID	Kelas
1	CARNATION Coffee-mate 495g
2	MILO ACTIVGO RTD 240ml
3	MILO ACTIVGO RTD HiCal 240ml
4	NESCAFE Eclair Latte PET 220ml
5	NESCAFE Honey Latte PET 220ml
6	NESCAFE ChocoBananaLatte PET 220ml
7	BEAR BRAND RTD Milk Tin 189ml
8	BEAR BRAND White Tea RTD Tin 140ml
9	BEAR BRAND White Malt RTD 140ml
10	DANCOW StrawberryFortigroUHT 110ml
11	DANCOW Coklat Fortigro UHT 110ml
12	CARNATION SCC 370g
13	NESCAFE Original Can 240ml
14	NESCAFE Latte Can 240ml
15	NESCAFE Mocha Can 240ml
16	MILO ACTIV-GO Nutriup 225ml
17	BEAR BRAND RTD Milk Tin (5(6x189ml)
18	DANCOW Strw Fortgr UHT 180ml
19	DANCOW Coklat Fortgr UHT 180ml
20	CARNATION Evaporated 405g
21	NESTLE GOODNES Kurma Milk UHT 180ml
22	NESTLE GOODNES Plain Milk UHT 180ml
23	FRISIAN FLAG

24	ULTRA
25	NESTLE NONA Plain SBC Can 370g
26	NESTLE GOODNES Kurma Milk 189ml
27	NESTLE GOODNES Kencur Milk UHT 180m
28	NEST GOODNES JaheMaduMilk UHT 180ml
29	MILO ACTIV-GO Nutriup 225ml
30	NESCAFE Coffee Cream UHT 180ml
31	NESCAFE Black UHT 180ml
32	INDOMILK
33	CAP ENAK
34	NESCAFE Kurma Latte PET 220ml
35	STARBUCKS Doubleshot Esp Latte 220ml
36	STARBUCKS Doubleshot Mocha 220ml
37	OMELA
38	NESCAFE Cappuccino 220ml
39	NESCAFE Latte 220ml
40	NESCAFE Caramel Macchiato 220ml
41	NESCAFE French Vanilla UHT 180ml
42	GOOD DAY
43	GOLDA COFFEE
44	POKKA
45	MILO ACTIV-GO UHT Cmbk 180ml
46	DEL MONTE
47	MILO ACTIV-GO UHT Cmbk 9(4x180ml)
48	HILO
49	DANCOW Coklat Fortigro UHT 110ml
50	MILO ACTIV-GO UHT Cmbk 110ml
51	MILO ACTIV-GO UHT Cmbk 9(110ml)
52	TUJUH KURMA
53	DANCOW Strw Fortgr UHT 180ml
54	DANCOW Coklat Fortgr UHT 180ml

55	DANCOW StrawberryFortigroUHT 110ml
56	DANCOW Coklat Fortigro UHT 9(110ml)
57	DANCOW Vanila Fortigro UHT 9(110ml)
58	DANCOW StrawberryFortigroUHT9(110ml)
59	NESCAFE Thai Milk Coffee 220ml
60	STARBUCKS Frappuccino Coffee 280ml
61	BEAR BRAND GOLDY RTD MILKTIN 189
62	NESCAFE Latte Can Cmbk 12(176+64ml)
63	GREENFIELDS
64	L-MEN
65	TIGA SAPI
66	CARNATION SBC Plain Tube 6(180g)
67	MILO ACTIVGO RTD 240ml
68	MILO ACTIV-GO UHT Cmbk 6(110ml)
69	MILO ACTIV-GO UHT Cmbk 9(180ml)
70	DANCOW Vanila Fortigro UHT6(110ml)
71	DANCOW Coklat FortigroUHT 6(110ml)
72	DANCOW Strw Fortigro UHT 6(110ml)
73	NESCAFE Ice Black 220ml
74	MILO ACTIV-GO Nutri Up RTD 220ml
75	NESCAFE Bis Koffee 220ml
76	NESTLE GOODNES KURMA AJWA MILK 189ML
77	MILO UHT Nutriactiv Cereal 180ml
78	MILO UHT Nutriactiv Banana 180ml
79	NESCAFE Ice Black 220ml
80	NESCAFE Ice Black Lychee 220ml
81	NESCAFE Latte 220ml
82	NESCAFE Caramel Macchiato 220ml
83	NESCAFE Cappuccino 220ml
84	DANCOW Vanila Fortigro UHT 110ml
85	DANCOW StrawberryFortigroUHT 110ml

86	DANCOW Coklat Fortigro UHT 110ml
87	DANCOW Vanila Fortigro UHT 110ml
88	DANCOW Plain Fortigro UHT 110ml
89	DANCOW Coklat Fortigro UHT 9(110ml)
90	BOOST OPTIMUM ACB011 400g
91	BOOST OPTIMUM ACB011 800g
92	NUTREN DIABETES ACD009 400g
93	NUTREN DIAB ACD009 400g
94	NUTREN JNR PREBIO 1 ACB003 400g
95	NUTREN JUNIOR ACB003 800g
96	PEPTAMEN JUNIOR ACE002-2 400g
97	PEPTAMEN ACE003 400g
98	ISOCAL BIB 400g
99	NARAYA
100	WRP

3.2 Pelabelan Data

Data yang dikumpulkan selanjutnya diberikan label terhadap objek-objek yang ada di dalam setiap data gambar. Proses pelabelan data dilakukan dengan software LabelImg. Setiap data gambar diberikan anotasi dengan memberikan bounding box terhadap objek produk yang ada pada gambar tersebut. Proses anotasi dalam penelitian ini disimpan dalam file dengan format PASCAL VOC XML. Contoh dari file XML pelabelan data dapat dilihat pada Gambar 3.4.

```
<annotation>
    <folder></folder>
    <filename>000001.jpg</filename>
    <path>000001.jpg</path>
    <source>
        <database>roboflow.ai</database>
    </source>
    <size>
        <width>500</width>
        <height>375</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
```

```
        <name>helmet</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
            <xmin>179</xmin>
            <xmax>231</xmax>
            <ymin>85</ymin>
            <ymax>144</ymax>
        </bndbox>
    </object>
    <object>
        <name>helmet</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
            <xmin>112</xmin>
            <xmax>135</xmax>
            <ymin>145</ymin>
            <ymax>175</ymax>
        </bndbox>
    </object>
</annotation>
```

Gambar 3. 4 Contoh Isi File XML dengan Format Anotasi PASCAL VOC

Sumber: roboflow.com

3.3 Pemrosesan Data

Pemrosesan data mencakup berbagai langkah untuk mempersiapkan data agar bisa digunakan secara efektif oleh model. Pemrosesan dilkakukan dengan memanfaatkan data yang telah dikumpulkan dan telah dianotasi. Tahap pemrosesan data pada penilitian ini dilakukan dalam beberapa tahap, mulai dari *data splitting*, *resizing*, dan pengubahan bentuk dataset ke dalam format TFRecord.

3.3.1 *Data Splittng*

Data yang ada kemudian dipisah menjadi data latih dan data uji. Proses ini dikenal dengan *data splitting*. Proses ini dilakukan untuk memastikan bahwa model yang dilatih memiliki kemampuan generalisasi yang baik. Data latih digunakan untuk melatih model agar dapat mengenali pola dari data, sementara data uji digunakan untuk mengevaluasi performa model pada data yang belum pernah dilihat sebelumnya. Tujuan utama dari *data splitting* adalah untuk menghindari *overfitting* dan memberikan evaluasi model yang objektif.

3.3.2 *Resizing*

Resizing gambar dilakukan dalam penelitian ini untuk memastikan konsistensi ukuran input yang sesuai dengan spesifikasi model deteksi objek. Selain itu, *resizing* juga membantu mengurangi kompleksitas komputasi dan penggunaan memori, sehingga memungkinkan pelatihan model yang lebih efisien dan inferensi yang lebih cepat, terutama pada perangkat berdaya rendah seperti smartphone. Proses *resizing* juga dioptimalkan agar gambar tetap dapat digunakan untuk deteksi objek tanpa mengorbankan akurasi secara signifikan.

3.3.3 TFRecord

Pada penelitian ini, gambar dalam format JPG, JPEG, PNG, dan jpg serta anotasi dalam format XML dikonversi ke dalam format TFRecord untuk mempermudah proses pelatihan model deteksi objek dengan TensorFlow. TFRecord dipilih karena format ini lebih efisien dalam hal penyimpanan dan akses data. Selain itu, format ini memungkinkan integrasi yang lebih baik dengan TensorFlow Object Detection API, sehingga mempercepat proses pelatihan dan inferensi. Proses konversi melibatkan pembacaan informasi gambar dan bounding box dari file XML, kemudian mengemasnya dalam bentuk format biner untuk disimpan dalam file TFRecord.

3.4 Pelatihan Data

Pelatihan data dilakukan terhadap dataset yang sudah berupa format TFRecord. Pada penelitian ini, pelatihan dilakukan sebanyak empat kali yakni dengan konfigurasi resize yang berbeda, yakni 320x320, 640x640, dan 1024x1024 untuk TensorFlow 2. Sedangkan untuk TensorFlow 1, digunakan konfigurasi resize sebesar 1024x1024. Pemilihan model TensorFlow 1 tersebut diambil berdasarkan percobaan sebelumnya yang pernah dilakukan oleh penulis.

Jumlah step yang dipakai untuk keempat model tersebut adalah sebesar 40.000. Step adalah iterasi atau siklus yang menggambarkan satu kali update bobot dan bias dalam suatu model. Pemilihan angka 40.000 step dalam penelitian ini tidak dimaksudkan untuk mencari angka step paling optimal, melainkan untuk memastikan kondisi pelatihan yang cukup baik untuk mencapai konvergensi awal. Angka ini dipilih berdasarkan pertimbangan efisiensi waktu pelatihan serta keseimbangan antara kualitas model dan keterbatasan sumber daya yang tersedia.

Batch size yang digunakan untuk ketiga model tersebut adalah sebesar 16. *Batch size* adalah jumlah gambar yang diproses dalam satu step. Jumlah batch size sebesar 16 dipilih

karena mempertimbangkan spesifikasi komputer yang telah disebutkan di awal Bab 3. Selain itu, angka batch size 16 juga cukup untuk menjaga keseimbangan antara stabilitas dan kovergensi. Untuk meningkatkan stabilitas pelatihan model, digunakan Batch Normalization dengan decay 0.97 untuk TensorFlow 1 dan 0.997 untuk TensorFlow 2. Sementara dropout tidak digunakan karena dataset yang besar sudah cukup untuk mendukung generalisasi tanpa khawatir akan terjadinya *overfitting*.

3.5 Integrasi Model

Pada penelitian ini, model deteksi objek yang telah dilatih menggunakan TensorFlow diintegrasikan dengan menggunakan TensorFlow Lite untuk memungkinkan penerapan di perangkat berdaya rendah seperti smartphone atau perangkat IoT. Proses integrasi ini melibatkan konversi model TensorFlow ke format tflite menggunakan TensorFlow Lite Converter, dengan opsi optimasi melalui teknik kuantisasi Dynamic Range Quantization untuk mempercepat inferensi dan mengurangi ukuran model. Dengan implementasi TensorFlow Lite, model dapat dijalankan secara efisien di perangkat lokal tanpa memerlukan koneksi internet, sehingga cocok untuk aplikasi deteksi objek secara real-time.

3.6 Evaluasi Model

Evaluasi model dalam penelitian ini menggunakan metrik *mean Average Precision* (mAP) untuk mengukur performa deteksi objek. Dalam penelitian ini, mAP diukur pada ambang batas IoU ≥ 0.5 (mAP@0.5), di mana prediksi dianggap benar jika tingkat tumpang tindih antara bounding box prediksi dan ground truth melebihi 50%. Penggunakan IoU dengan threshold 0.5 dipilih karena angka tersebut merupakan salah satu metrik yang paling umum digunakan dalam evaluasi model deteksi objek, terutama dalam dataset seperti COCO dan Pascal VOC. Hal ini membuat hasil penelitian lebih mudah dibandingkan dengan penelitian lain yang menggunakan standar yang sama. Pada tahap ini, evaluasi dilakukan terhadap dua skenario, yakni evaluasi terhadap model asli dan model tflite.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Pengumpulan data dilakukan dengan mengikuti kriteria yang sudah dijabarkan pada Subbab 3.1.1. Jumlah kelas dari data yang digunakan adalah sebanyak 100 kelas. Total foto yang dikumpulkan dalam tahap ini adalah sebanyak 10.040. Pengambilkan foto dilakukan di minimarket atau toko-toko yang memiliki stok produk yang terdaftar pada *Stock Keeping Unit* (SKU) untuk dideteksi seperti pada Tabel 3.1. Contoh foto dapat dilihat pada Gambar 4.1 dan Gambar 4. 2.

Gambar 4. 1 Contoh Pengambilan Gambar pada Produk di Rak Gondala

Pada Gambar 4. 1, dapat dilihat bahwa pengambilan gambar dilakukan langsung pada minimarket di rak gondala. Pada gambar tersebut, terdapat beberapa SKU yang menjadi target deteksi seperti BEAR BRAND dan GOODNES. Untuk contoh pengambilan gambar pada showcase cooler dapat dilihat pada Gambar 4. 2. Pada gamber tersebut, terdapat beberapa SKU target yang terlihat pada showcase cooler seperti NESCAFE dan STARBUCK.

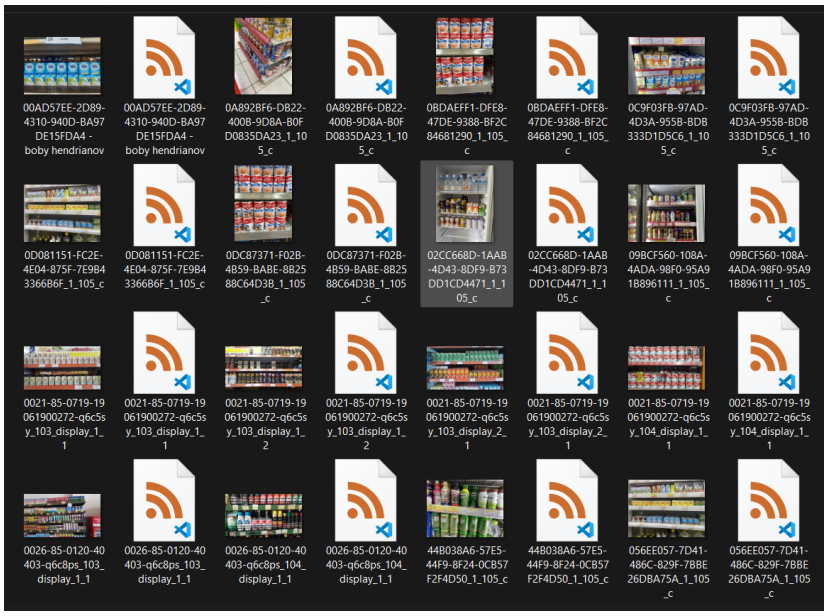
Gambar 4. 2 Contoh Pengambilan Gambar pada Produk di Showcase Cooler

4.2 Pelabelan Data

Pelabelan data dilakukan terhadap seluruh gambar yang sudah dikumpulkan pada tahap pengumpulan data. Seperti yang sudah dijelaskan pada subbab 3.2, proses pelabelan data dilakukan dengan memanfaatkan tools bernama LabelImg. Berdasarkan data gambar yang didapatkan, jumlah rata-rata objek adalah sebelas objek per foto dan jumlah rata-rata kelas adalah tiga kelas per foto. Contoh proses pelabelan data dapat dilihat pada Gambar 4. 3.

Pada proses pelabelan data, setiap selesai menganotasi satu gambar, maka LabelImg secara otomatis menyimpan satu file XML dengan penamaan yang sama seperti nama file Gambar tersebut. Sehingga, pada saat selesai menganotasi keseluruhan gambar, maka jumlah file XML yang dihasilkan akan sama dengan jumlah gambar yang ada pada dataset. Contoh output file XML yang dihasilkan pada proses pelabelan data dapat dilihat pada Gambar 4. 4 dan untuk isi file XML setelah proses anotasi dapat dilihat pada Gambar 4. 5.

Gambar 4. 3 Contoh Proses Pelabelan Data dengan LabelImg



Gambar 4. 4 File XML yang dihasilkan pada proses Pelabelan Data

Pada file XML yang dihasilkan, terdapat berbagai tag yang menyimpan berbagai informasi yang terkait dengan anotasi data. Beberapa tag yang penting dalam menyimpan informasi anotasi antara lain: tag <annotation> yang menandakan bahwa file tersebut merupakan file dengan format PASCAL VOC XML, tag <path > yang munjukkan lokasi gambar yang dianotasi, tag <size> yang menyimpan informasi ukuran dari gambar, dan tag <object> yang menyimpan informasi objek yang dituju. Pada tag <object>, informasi nama objek yang

diletakkan pada tag <name> dan lokasi dari bounding box objek yang diletakkan pada tag <bndbox>. Contoh isi dari file XML dapat dilihat pada Gambar 4.5.

```
<annotation>
    <folder>FRISIAN FLAG</folder>
    <filename>gambar.jpeg</filename>
    <path>D:\gambar.jpeg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1024</width>
        <height>768</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>FRISIAN FLAG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>12</xmin>
            <ymin>350</ymin>
            <xmax>136</xmax>
            <ymax>645</ymax>
        </bndbox>
    </object>
    <object>
        <name>FRISIAN FLAG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>151</xmin>
            <ymin>351</ymin>
            <xmax>286</xmax>
            <ymax>648</ymax>
        </bndbox>
    </object>
    <object>
        <name>FRISIAN FLAG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>291</xmin>
            <ymin>352</ymin>
            <xmax>426</xmax>
            <ymax>650</ymax>
        </bndbox>
    </object>
    <object>
        <name>FRISIAN FLAG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
```

```
        <bndbox>
            <xmin>438</xmin>
            <ymin>348</ymin>
            <xmax>582</xmax>
            <ymax>651</ymax>
        </bndbox>
    </object>
    <object>
        <name>FRISIAN FLAG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>586</xmin>
            <ymin>346</ymin>
            <xmax>730</xmax>
            <ymax>649</ymax>
        </bndbox>
    </object>
    <object>
        <name>FRISIAN FLAG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>743</xmin>
            <ymin>339</ymin>
            <xmax>880</xmax>
            <ymax>648</ymax>
        </bndbox>
    </object>
    <object>
        <name>FRISIAN FLAG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>886</xmin>
            <ymin>343</ymin>
            <xmax>1023</xmax>
            <ymax>652</ymax>
        </bndbox>
    </object>
</annotation>
```

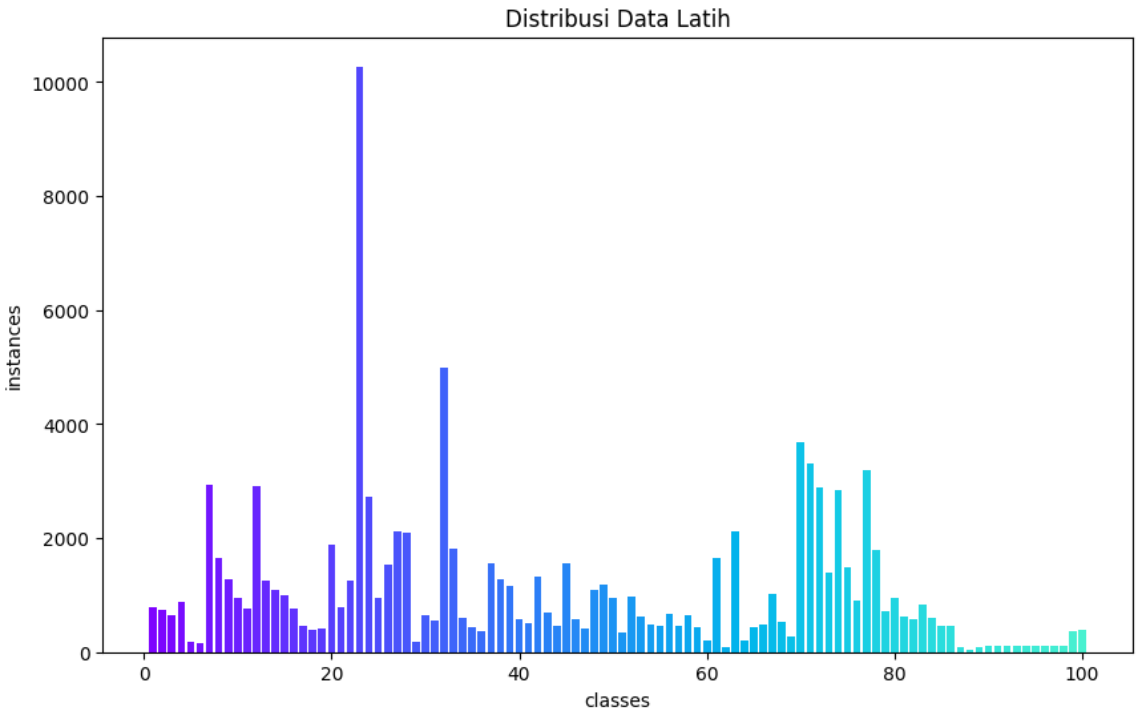
Gambar 4. 5 Contoh Anotasi Gambar berupa File XML

4.3 Pemrosesan Data

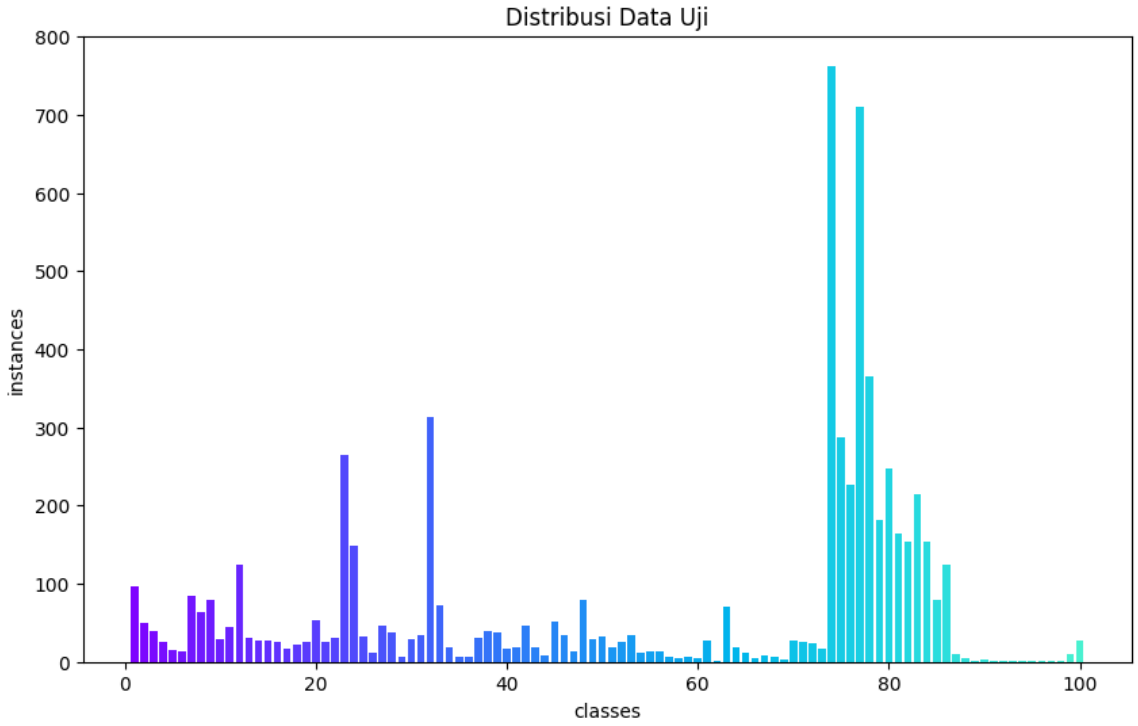
Pemrosesan data dilakukan dengan berbagai tahapan dimulai dengan tahap *resizing*, yakni pengubangan ukuran gambar kepada bentuk yang sesuai dengan size input model. Ukurang gambar yang didapatkan saat proses pengupulan data sangat bervariasi, mulai dari 1024x786 sebagai ukuran paling dominan, kemudian ada ukuran 1024x576, 1024x1024, dan berbagai ukuran lainnya. Oleh sebab itu, perlu dilakukan *resizing* agar dapat diproses oleh

model. Tiga ukuran resize yang digunakan dalam penelitian ini adalah 320x320, 640x640, dan 1024x1024.

Kemudian untuk *data splitting*, dataset yang ada dipisah menjadi data latih sebanyak 9.371 gambar dengan 106.555 objek, dan data uji sebanyak 669 gambar dengan 6.567 objek. Pembagian ini mengikuti proporsi standar dalam pembelajaran mesin, yaitu sekitar 93% untuk data latih dan 7% untuk data uji. Jumlah perbandingan ini digunakan karena dengan jumlah data yang cukup banyak, proporsi 7% untuk data pelatihan sudah cukup untuk mewakili gambar pengujian. Selain itu, dalam skenario dunia nyata dibutuhkan jumlah data pelatihan yang lebih banyak sehingga menghasilkan model prediksi yang lebih akurat. Setelah seluruh data terbagi menjadi data latih dan data uji, format dari data tersebut diubah ke dalam format TFRecord. Distribusi data yang digunakan dalam proses ini dapat dilihat pada Gambar 4. 6 dan Gambar 4. 7.



Gambar 4. 6 Distribusi Kelas Data Latih



Gambar 4. 7 Distribusi Kelas Data Uji

4.4 Pelatihan Data

Proses pelatihan data dilakukan sebanyak empat kali dengan menggunakan arsitektur SSD MobileNetV2 FPNLite untuk TensorFlow 2 dan SSD MobileNetV3 Large untuk TensorFlow 1, sesuai dengan yang sudah disebutkan pada subbab 3.3. Setiap pelatihan dilakukan dengan ukuran input gambar yang berbeda, yaitu 320x320, 640x640, dan 1024x1024 piksel untuk TensorFlow 2 dan 1024x1024 piksel untuk TensorFlow 1. Perbedaan ukuran ini bertujuan untuk mengevaluasi bagaimana resolusi gambar memengaruhi kinerja model dalam mendeteksi objek, baik dari segi akurasi maupun waktu pelatihan. Proses pelatihan untuk TensorFlow 2 dilakukan dengan menjalankan skrip seperti yang terlihat pada Gambar 4. 8, sedangkan pelatihan untuk TensorFlow 1 dilakukkan dengan menjalankan skrip seperti yang terlihat pada Gambar 4. 9.

```
CUDA_VISIBLE_DEVICES=0,1 python model_main_tf2.py \
    --pipeline_config_path=./c5_31_320x320/pipeline.config \
    --model_dir=./c5_31_320x320/ \
    --alsologtostderr \
    --num_train_steps=40000 \
    --sample_1_of_n_eval_examples=1
```

Gambar 4. 8 Perintah untuk Memulai Pelatihan Model pada TensorFlow 2.

Pada Gambar 4. 8 Perintah untuk Memulai Pelatihan Model pada TensorFlow 2., terdapat perintah CUDA_VISIBLE_DEVICES=0,1 menandakan penggunaan GPU yang dipakai, yakni GPU 0 dan GPU 1. Dengan perintah tersebut, pelatihan dilakukan dengan menggunakan dua buah GPU Nvidia Tesla V100 32 GB. Penggunaan dua buah GPU dilakukan agar mesin mampu menangani pelatihan gambar dalam batch yang lebih besar dan dapat meningkatkan waktu pelatihan. Selain itu, terdapat perintah "python model_main_tf2.py". Perintah tersebut merupakan perintah dalam command linux yang digunakan untuk menjalankan skrip python yakni model_main_tf2.py. File python ini tersedia dalam repository TensorFlow Object Detection API. Dalam perintah tersebut, ditambahkan juga beberapa flag seperti flag --pipeline_config_path yang menunjukkan lokasi file config dan flag --model_dir yang menunjukkan lokasi model pelatihan yang akan disimpan.

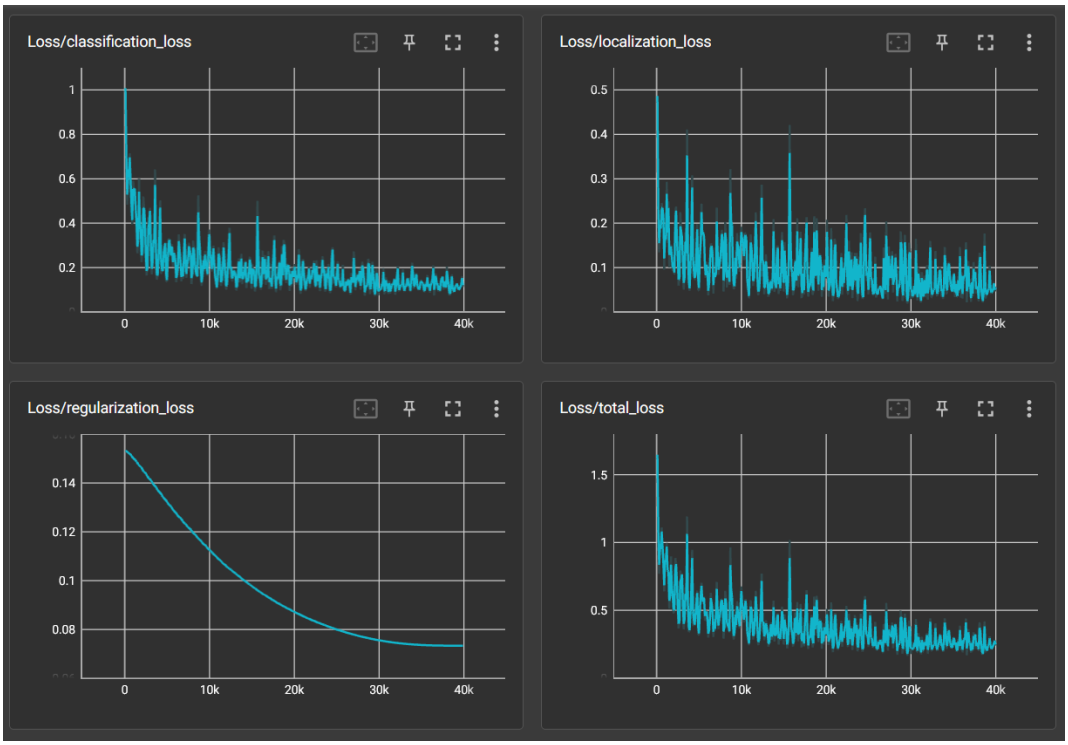
```
CUDA_VISIBLE_DEVICES=2,3 python train.py \
    --logtostderr \
    --train_dir=./models/ c5_31_tf1/train \
    --pipeline_config_path=./models/c5_31_tf1/pipeline.config \
    --ps_tasks=1 \
    --num_clones=2
```

Gambar 4. 9 Perintah untuk Memulai Pelatihan Model pada TensorFlow 1.

Ada sedikit perbedaan antara perintah memulai pelatihan antara TensorFlow 2 dengan TensorFlow 1 seperti yang terlihat pada Gambar 4. 9. Pada perintah pelatihan di TensorFlow 1, terdapat juga perintah CUDA_VISIBLE_DEVICES=2,3 yang menandakan pelatihan menggunakan dua buah GPU, yakni GPU 2 dan GPU 3. Kemudian, perintah "python train.py" merupakan perintah untuk menjalankan skrip python bernama train.py yang tersedia dalam repository TensorFlow Object Detection API. Terdapat pula beberapa flags yang berbeda dengan perintah pelatihan TensorFlow 2 seperti flag --train_dir yang menunjukkan lokasi penyimpanan model dan flag –num_clones yang menunjukkan jumlah GPU yang digunakan.

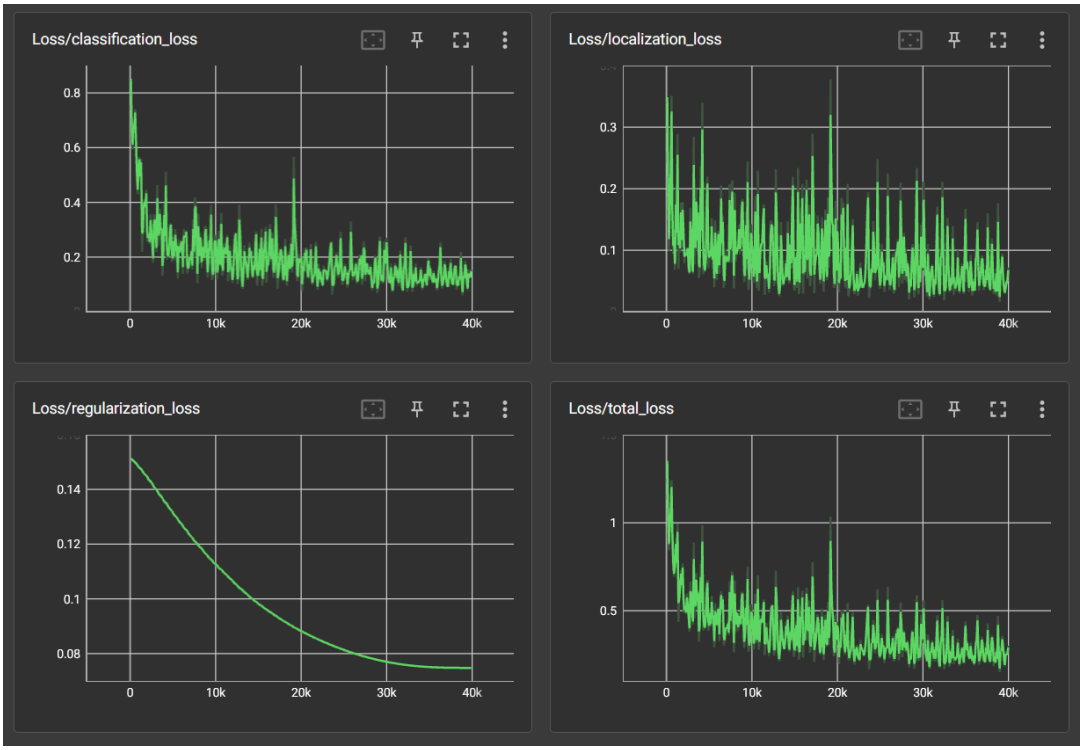
Pada saat proses pelatihan, terdapat informasi yang disimpan seperti classification loss, localization loss, dan total loss dari setiap step pelatihan. Informasi-informasi tersebut dapat

disajikan dengan visualisasi grafik dengan memanfaatkan salah satu tools yang tersedia dari TensorFlow yakni TensorBoard. Grafik pelatihan untuk SSD MobileNetV2 FPNLite 320x320 dapat dilihat pada Gambar 4. 10, grafik pelatihan untuk SSD MobileNetV2 FPNLite 640x640 dapat dilihat pada Gambar 4. 11, grafik pelatihan untuk SSD MobileNetV2 FPNLite 1024x1024 dapat dilihat pada Gambar 4. 12, dan grafik pelatihan untuk SSD MobileNetV3 Large 1024x1024 dapat dilihat pada Gambar 4. 13.

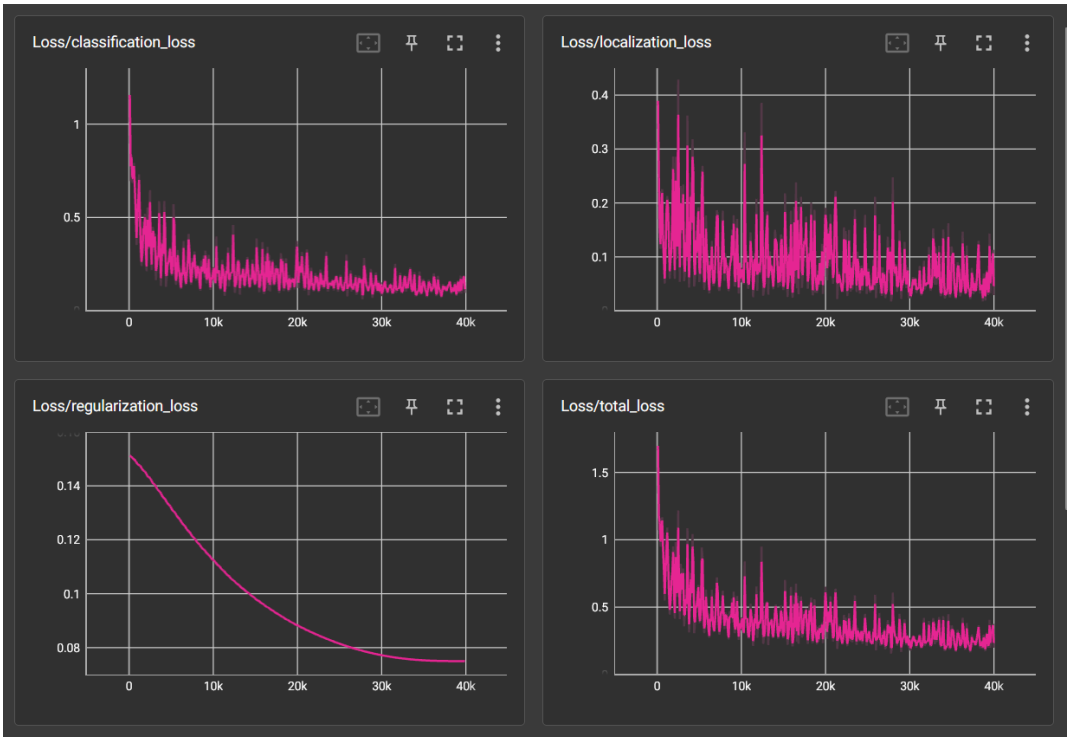


Gambar 4. 10 Grafik Pelatihan untuk Model 320x320 pada TensorFlow 2

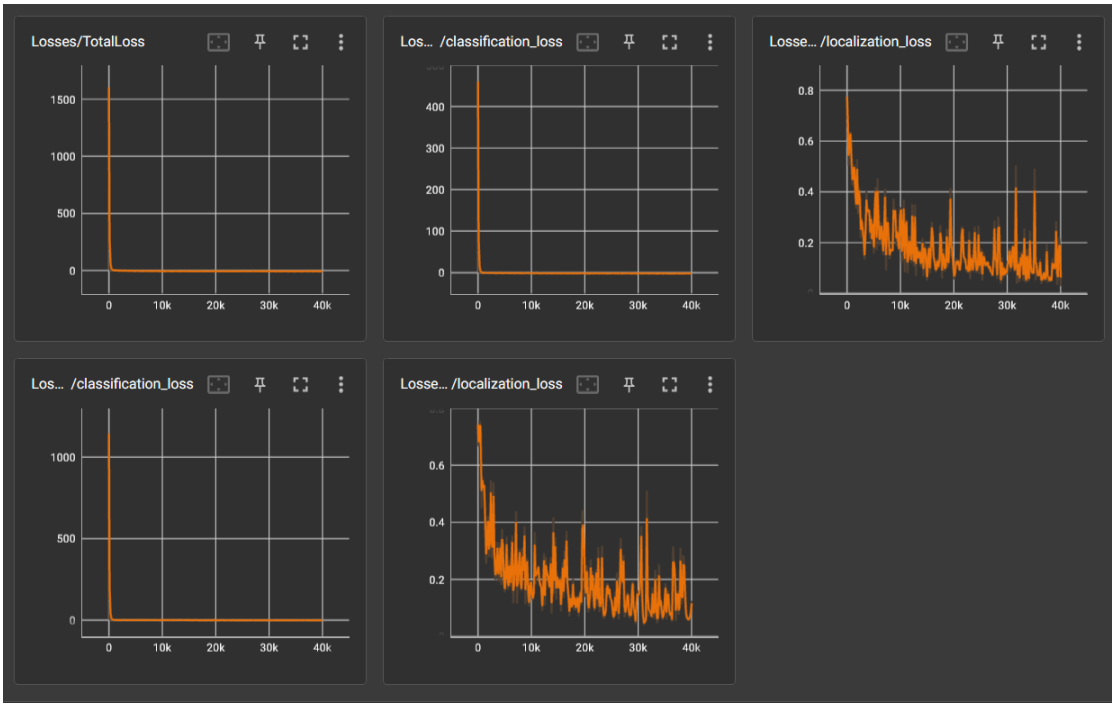
Grafik Tensorboard yang ditunjukkan oleh hasil pelatihan di TensorFlow 2 cenderung sama untuk ketiga model, yakni model 320x320, model 640x640, dan model 1024x1024. Hal itu dapat dilihat pada Gambar 4. 10, Gambar 4. 11, dan Gambar 4. 12. Ketiga model tersebut memiliki grafik classification loss, localization loss, dan total loss yang cenderung naik turun. Sedangkan pada TensorBoard di TensorFlow1, grafik classification loss cenderung menurun tajam di awal pelatihan dan sidikit menurut secara stabil sehingga berpengaruh juga terhadap total loss yang stabil. Hal itu dapat dilihat pada Gambar 4. 13. Persamaan antara grafik TensorFlow 1 dan 2 adalah pada grafik localization loss di mana keduanya memiliki grafik yang cenderung naik turun.



Gambar 4. 11 Grafik Pelatihan untuk Model 640x640 pada TensorFlow 2



Gambar 4. 12 Grafik Pelatihan untuk SSD Model 1024x1024 pada TensorFlow 2



Gambar 4. 13 Grafik Pelatihan untuk SSD Model 1024x1024 pada TensorFlow 1

Untuk TensorFlow 2 dengan arsitektur SSD MobileNetV2 FPNLite, model dengan resolusi input 320x320 memerlukan waktu pelatihan yang paling singkat, yaitu 1 jam 35 menit dan 41 detik, model dengan resolusi input 640x640 memerlukan waktu pelatihan selama 5 jam 23 menit dan 21 detik, dan model dengan resolusi input 1024x1024 memerlukan waktu yang paling lama, yaitu 12 jam 16 menit dan 21 detik. Untuk TensorFlow 1 dengan arsitektur SSD MobileNetV3 Large dan resolusi input 1024x1024 memerlukan waktu 7 jam 27 menit dan 39 detik. Durasi tersebut lebih lama daripada model TensorFlow 2 dengan resolusi input 640x640 namun lebih cepat daripada model TensorFlow 2 dengan resolusi input 1024x1024. Rangkuman proses pelatihan seluruh model dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Rangkuman Pelatihan Seluruh Model Penelitian.

TensorFlow Version	Model	Resizer	Jumlah Parameter	Training Time
2	SSD MobileNetV2 FPNLite	320x320	5309234	1:35:41
2	SSD MobileNetV2 FPNLite	640x640	5309234	5:23:08
2	SSD MobileNetV2 FPNLite	1024x1024	5309234	12:16:21
1	SSD MobileNetV3 Large	1024x1024	6766275	7:37:39

Peningkatan waktu pelatihan disebabkan oleh bertambahnya jumlah piksel yang harus diproses oleh model pada setiap gambar, sehingga menambah kompleksitas komputasi. Hal ini menggambarkan adanya *trade-off* antara resolusi gambar dan efisiensi waktu pelatihan. Resolusi yang lebih tinggi memungkinkan model untuk menangkap lebih banyak detail dari gambar, namun di sisi lain, membutuhkan waktu pelatihan yang lebih lama.

Informasi lain yang didapatkan dari Tabel 4.1 adalah perbedaan jumlah parameter antara arsitektur SSD MobileNetV3 Large dengan SSD MobileNetV2 FPNLite. Hal ini disebabkan MobileNetV3 menggunakan optimisasi block untuk meningkatkan fitur yang relevan terhadap objek sehingga backbone yang dimiliki lebih besar dibandingkan dengan MobileNetV2. Backbone yang dimiliki MobileNetV3 ini meningkatkan akurasi deteksi objek walaupun secara efisiensi model berada di bawah MobileNetV2.

4.5 Integrasi Model

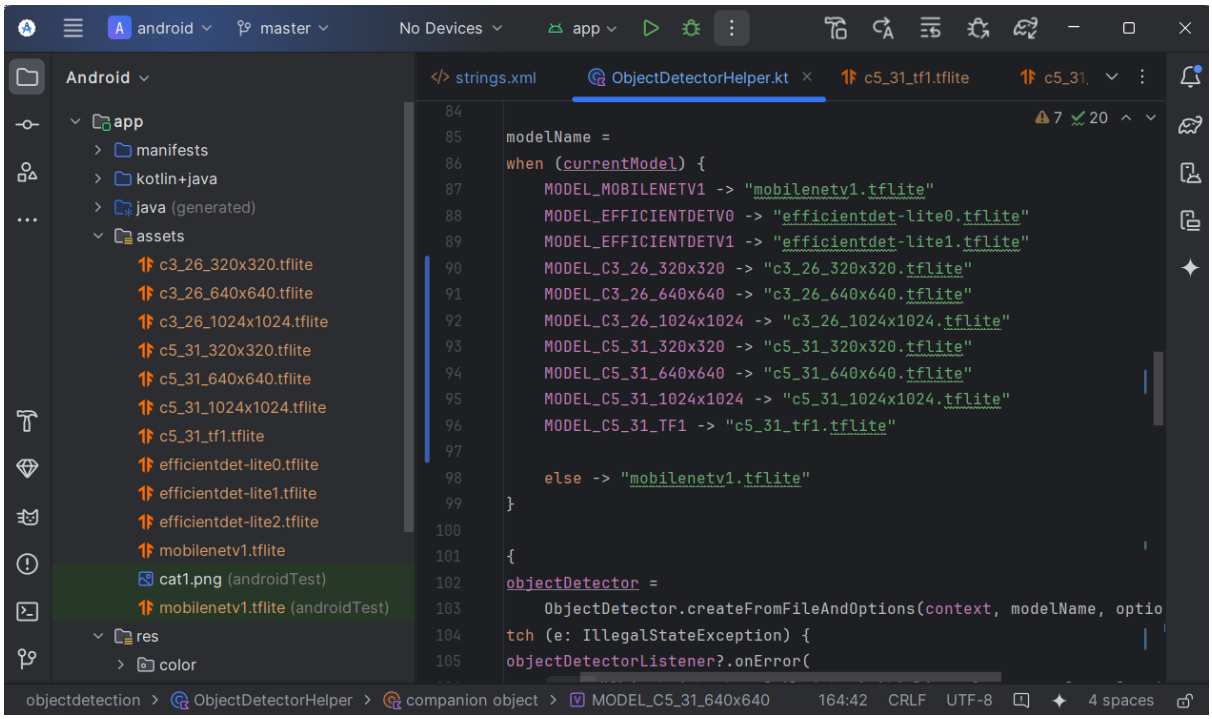
Integrasi model dilakukan dengan mengubah format model ke dalam bentuk TensorFlow Lite (TFLite) agar dapat dijalankan di perangkat dengan sumber daya terbatas, seperti perangkat mobile atau edge devices. Proses konversi TFLite untuk model TensorFlow 1 dilakukan secara langsung tanpa menggunakan kuantisasi karena pada pengujian terdahulu peneliti, model TensorFlow 1 dapat berjalan dengan baik tanpa perlu menggunakan kuantisasi. Untuk model dengan TensorFlow 2, proses konversi menggunakan metode Dynamic Range Quantization, di mana bobot (weights) dari model dikompresi ke format yang lebih efisien, sementara aktivasi (activations) tetap menggunakan float32. Skrip untuk mengubah model pelatihan ke dalam format TFLite dapat dilihat pada Gambar 4. 14.

```
import tensorflow as tf

converter =
tf.lite.TFLiteConverter.from_saved_model('./training/ssd_mobilenet_v2_fpnli
te_320x320_coco17_tpu-8/c5_31_640x640/tflite/saved_model')
converter.allow_custom_ops = True

converter.optimizations = [tf.lite.Optimize.DEFAULT]

tflite_model = converter.convert()

with open('./training/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-
8/c5_31_640x640/tflite/detect_drq.tflite', 'wb') as f:
  f.write(tflite_model)
```

Gambar 4. 14 Skrip untuk Mengubah Model Pelatihan Menjadi Format TFLite

Dynamic Range Quantization bekerja dengan mengkuantisasi bobot model dari float32 ke int8 secara dinamis pada saat inferensi, tanpa perlu melakukan perubahan pada aktivasi. Dengan pendekatan ini, ukuran model dapat berkurang secara signifikan, memungkinkan penyimpanan dan pengolahan model yang lebih cepat, terutama di perangkat dengan memori dan daya komputasi terbatas. Keuntungan lain dari metode ini adalah proses inferensi menjadi lebih hemat daya karena bobot yang digunakan lebih kecil, tetapi akurasi tetap mendekati model float32 aslinya karena aktivasi tetap dipertahankan dalam format float32. Dynamic Range Quantization dipilih karena memberikan keseimbangan yang baik antara pengurangan ukuran model dan akurasi. Pendekatan ini memungkinkan model tetap dapat berfungsi secara optimal di perangkat target tanpa mengorbankan performa deteksi.

Setelah pengubahan format, model TFLite dapat diintegrasikan ke dalam kode program aplikasi android menggungakan Android Studio. Aplikasi android yang dipakai dalam penelitian ini adalah aplikasi yang disediakan oleh repository TensorFlow Object Detection API. Sebelum diintegrasikan, model TFLite diberikan metadata terlebih dahulu agar dapat dibaca oleh kode aplikasi android. Proses peletakkan model TFLite ke dalam aplikasi android dapat dilihat pada Gambar 4. 15.



Gambar 4. 15 Integrasi Model TFLite ke dalam Aplikasi Android dengan Android Studio

4.6 Evaluasi Model

Evaluasi model dilakukan terhadap dataset pengujian yang sudah disiapkan pada tahap pemrosesan data. Data pengujian terdiri dari 669 gambar dan 6567 objek. Evaluasi dilakukan dengan memanfaatkan sumber daya komputer sesuai spesifikasi yang tertera pada Bab 3. Hasil dari evaluasi dapat dilihat pada Tabel 4. 2.

Tabel 4. 2 Hasil Evaluasi Seluruh Model Penelitian

TensorFlow Version	Model	Localization Loss	Classification Loss	mAP@0.5IOU
2	SSD MobileNetV2 FPNLite 320x320	0.084647	0.208833	0.782356
2	SSD MobileNetV2 FPNLite 640x640	0.071133	0.182388	0.813912
2	SSD MobileNetV2 FPNLite 1024x1024	0.069394	0.176482	0.81718
1	SSD MobileNetV3 Large 1024x1024	0.092231	0.378762	0.729781

Pada evaluasi dengan komputer, ketiga model TensorFlow 2 mendapatkan nilai loss dan mAP yang hampir sama. Sedangkan model TensorFlow 1 mendapatkan nilai loss yang lebih tinggi dan skor mAP yang lebih rendah dibandingkan ketiga model TensorFlow 2. Model TensorFlow 1 dengan ukuran input sebesar 1024x1024 mendapatkan skor mAP terendah, yakni sebesar 0.729781. Untuk TensorFlow 2, model dengan ukuran input 320x320 mendapatkan skor mAP sebesar 0.782356, model 640x640 mendapatkan skor mAP sebesar 0.813912, dan model 1024x1024 mendapatkan skor mAP yang paling besar, yakni sebesar 0.813912. Hasil inferensi gambar pada perangkat komputer dapat dilihat pada Gambar 4. 16.

Terdapat juga hasil evaluasi AP untuk setiap kelasnya. Hasil evaluasi ini penting untuk mengetahui kelas mana saja yang memiliki hasil evaluasi yang baik dan yang buruk. Dengan demikian, proses analisis terhadap kelemahan model menjadi lebih mudah khususnya terhadap kelas yang memiliki hasil evaluasi yang masih rendah. Hasil evaluasi untuk setiap label dapat dilihat pada Lampiran.

Hasil evaluasi yang didapatkan oleh setiap kelas berbeda antara satu dengan yang lainnya. Kelas yang memiliki jumlah objek sedikit dalam dataset pengujian memiliki kecenderungan mendapat nilai AP yang tinggi karena seluruh objek yang ada terdeteksi oleh model. Hal ini dapat dilihat pada SKU NUTREN dan BOOST OPTIMUM. Masing-masing dari SKU tersebut hanya memiliki dua objek dalam data evaluasi dan keduanya mendapatkan skor AP sempurna yakni satu. Kelas yang memiliki nilai AP rendah umumnya merupakan kelas

yang tidak memiliki gramasi dalam penamaan SKU-nya. Hal ini disebabkan objek kelas tersebut memiliki gramasi dan desain yang berbeda-beda sehinga model lebih sulit untuk mengenali pola yang ada.

Gambar 4. 16 Kiri: Ground Truth. Kanan: Hasil Inferensi Model 320x320

Pada Gambar 4. 16, terdapat perbandingan antara ground truth di sebelah kiri dan hasil inferensi dari model 320x320 di sebelah kanan. Hasil prediksi dari model 320x320 menunjukkan hasil yang sesuai dengan bounding box yang ada pada ground truth. Model yang dihasilkan dari hasil pelatihan sudah cukup baik dalam mendeteksi objek pada dataset uji. Namun, perlu juga dilakukan evaluasi terhadap model TFLite agar hasil evaluasi lebih representatif terhadap studi kasus deteksi objek pada perangkat smartphone.

Pada evaluasi TFLite, parameter yang dijadikan evaluasi adalah model size, inference time, dan mAP. Model size penting menjadi parameter evaluasi karena mempengaruhi ukuran dari aplikasi yang digunakan pada perangkat bergerak atau smartphone. Inference time juga menjadi perhatian karena model TFLite perlu memiliki waktu prediksi yang cepat dengan sumber daya terbatas. Terakhir, mAP juga masih menjadi parameter pengukuran akurasi seperti pada model asli. Hasil evaluasi dari model TFLite dapat dilihat pada Tabel 4. 3.

Tabel 4. 3 Hasil Evaluasi Model TFLite

TensorFlow Version	Model	Model Size (Megabytes)	Inference Time (Milliseconds)	mAP@0.5IOU
2	SSD MobileNetV2 FPNLite 320x320	3.44 MB	132 ms	0.894257511213198
2	SSD MobileNetV2 FPNLite 640x640	4.05 MB	413 ms	0.91543842821496
2	SSD MobileNetV2 FPNLite 1024x1024	4.33 MB	1039 ms	0.92092846480901
1	SSD MobileNetV3 Large 1024x1024	13.8 MB	395 ms	0.9017169316

Pada Tabel 4. 3, terdapat tabel yang berisi informasi terkait ukuran model, waktu inferensi, dan skor mAP. Untuk ukuran model pada TensorFlow 2, SSD MobileNetV2 FPNLite 320x320 memiliki ukuran file yang paling kecil yakni 3.44 MB. SSD MobileNetV2 FPNLite 640x640 memiliki ukuran file yang sedikit lebih besar, yakni 4.05 MB. SSD MobileNetV2 FPNLite 1024x1024 memiliki ukuran file yang paling besar diantara seluruh model TensorFlow 2 lainnya, dengan 4.33 MB. Namun, model TensorFlow 1 memiliki ukuran file yang jauh lebih besar dibandingkan dengan model TensorFlow 2, yakni dengan 13.8 MB.

Walaupun memiliki ukuran file yang paling besar, model TFLite TensorFlow 1 dengan ukuran input 1024x1024 memiliki waktu inferensi yang termasuk cepat dengan 395 ms. Model TensorFlow 2 dengan ukuran input 1024x1024 hanya mampu mendapatkan waktu inferensi sebesar 1038 ms, jauh lebih lama dibandingkan dengan model TensorFlow 1. Untuk ukuran lainnya, model TensorFlow 2 dengan ukuran input 640x640 memiliki waktu inferensi sebesar 413 ms dan model TensorFlow 2 dengan ukuran input 320x320 memiliki waktu inferensi sebesar 132 ms.

Dari segi akurasi, keempat model yang diuji dalam penelitian ini mendapatkan skor mAP yang tidak jauh berbeda. Model SSD MobileNetV3 Large 1024x1024 pada TensorFlow 1 mendapatkan skor mAP sebesar 0.9017169316. Untuk model pada TensorFlow 2, SSD MobileNetV2 FPNLite 320x320 mendapatkan skor mAP sebesar 0.894257511213198, SSD MobileNetV2 FPNLIte 640x640 mendapatkan skor mAP sebesar 0.91543842821496, dan SSD MobileNetV2 FPNLIte 1024x1024 mendapatkan skor mAP sebesar 0.92092846480901.

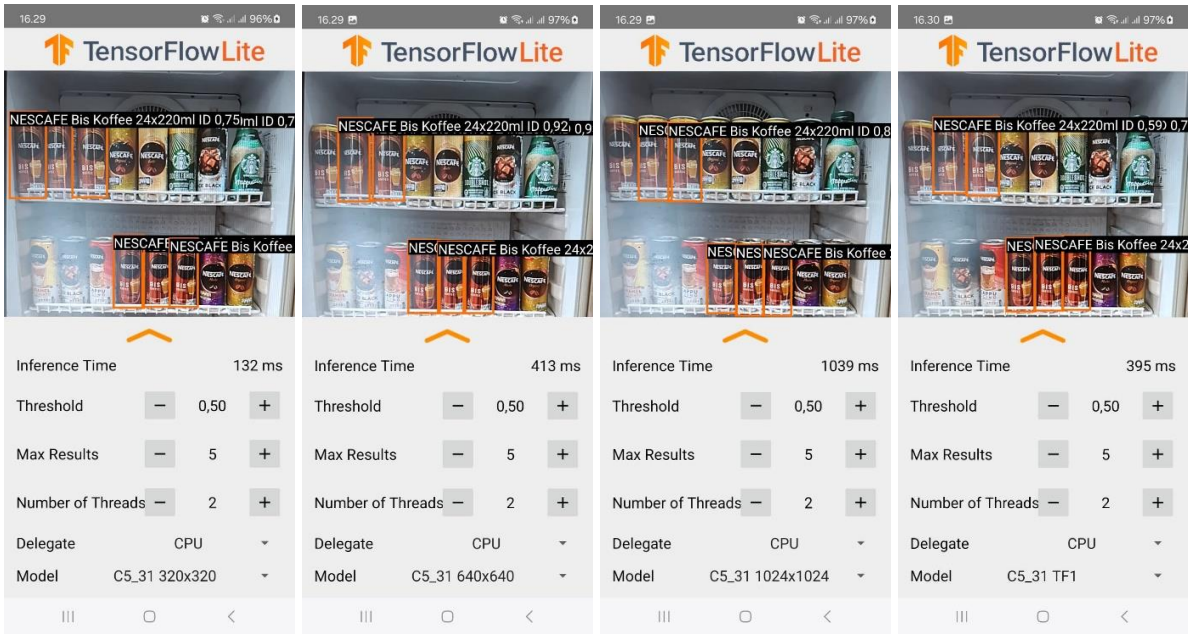
4.7 Pembahasan

Berdasarkan hasil yang didapatkan, dapat dilihat bahwa model SSD MobileNetV2 FPNLite pada TensorFlow 2 dan SSD MobileNetV3 Large pada TensorFlow 1 dapat terimplementasi dalam perangkat bergerak dengan memanfaatkan format TFLite. Perbedaan variasi dari resizer model, baik dari TensorFlow 1 maupun TensorFlow 2, memberikan perbedaan yang cukup signifikan pada waktu inferensi seperti yang dapat dilihat pada Tabel 4. 3. Pada evaluasi model asli di perangkat komputer, model TensorFlow 1 juga mendapatkan hasil evaluasi model yang cenderung lebih rendah dari model TensorFlow 2. Namun, keempat model tidak memiliki perbedaan yang cukup signifikan pada hasil evaluasi mAP setelah diubah ke dalam format TFLite.

Seluruh model mendapatkan peningkatan skor mAP saat diubah ke dalam format TFLite. Untuk SSD MobileNetV2 FPNLite di TensorFlow 2, model 320x320 mendapatkan skor mAP sebesar 0.782356 pada model asli dan naik menjadi 0.894257511213198 saat diubah ke dalam format TFLite. Model 640x640 medapatkan skor mAP sebesar 0.813912 pada model asli dan naik menjadi 0.91543842821496 pada format TFLite, dan model 1024x1024 mendapatkan skor mAP sebesar 0.81718 pada model asli kemudian naik menjadi 0.92092846480901 pada format TFLite. Untuk SSD MobileNetV3 Large di TensorFlow 1, model mendapatkan skor mAP sebesar 0.729781 pada model asli dan naik menjadi 0.9017169316 pada format TFLite.

Contoh pada Gambar 4. 17 Hasil Inferensi TFLite di Perangkat Bergerak. menunjukkan bahwa keempat model dapat mendeteksi objek-objek yang berada dalam rak dengan baik. Threshold yang digunakan adalah sebesar 0.5 dan jumlah objek maksimal yang dideksi oleh aplikasi adalah sebanyak 5 objek. Untuk perbandingan yang lebih komprehensif, inferensi dilakukan dengan memanfaatkan inferensi TFLite pada komputer menggunakan treshhold 0.5 dan menggunakan sampel gambar yang sama.

Gambar sampel yang sudah diberi ground truth bounding box dapat dilihat pada Gambar 4.18. Pada gambar tersebut, terdapat total 38 bounding box ground truth. Pada rak pertama, terdapat 9 objek SKU NESCAFE Kotak yang terdiri dari 5 Kotak NESCAFE Coffe Cream dan 4 Kotak NESCAFE Black. Pada rak kedua, terdapat 6 SKU MILO Botol yang terdiri 5 Botol MILO Nutriup Lama dan 1 Botol MILO Nutriup Baru. Pada rak ketiga dan keempat, terdapat 23 SKU MILO Kaleng yang terdiri dari 11 Kaleng MILO Activgo Calcion pada rak ketiga dan 12 Kaleng MILO Activgo Original pada rak keempat.



Gambar 4. 17 Hasil Inferensi TFLite di Perangkat Bergerak.
(Paling Kiri: Model TF 2 320x320; Kedua dari Kiri: Model TF 2 640x640;
Kedua dari Kanan: Model TF 2 1024x1024; Paling Kanan: Model TF 1 1024x1024)

Gambar 4. 18 Gambar Sampel beserta Ground Truth Bounding Box-nya.

Hasil dari inferensi Model TFLite pada SSD MobileNetV3 Large 1024x1024di TensorFlow 1 dapat dilihat pada Gambar 4. 19. Pada gambar tersebut, model mampu memprediksi seluruh SKU MILO Kaleng dan seluruh SKU NESCAFE dengan baik pada rak pertama, ketiga, dan keempat. Namun, terdapat false negative atau objek yang tidak terprediksi pada rak kedua. Objek-objek yang tidak terdeteksi yaitu SKU MILO Nutriup Baru dan satu dari lima SKU MILO Nutriup Lama.

Gambar 4. 19 Hasil dari inferensi Model TFLite 1024x1024 di TensorFlow 1

Hasil dari inferensi Model TFLite pada SSD MobileNetV2 FPNLite 320x320 di TensorFlow 2 dapat dilihat pada Gambar 4.20. Pada gambar tersebut, model mampu memprediksi seluruh SKU pada rak pertama dan kedua. Namun, model ini hanya mampu memprediksi 4 objek dari 11 objek ground truth yang ada pada rak ketiga. Model ini belum mampu memprediksi satupun objek yang ada pada rak keempat.

Hasil dari inferensi Model TFLite pada SSD MobileNetV2 FPNLite 640x640 di TensorFlow 2 dapat dilihat pada Gambar 4. 21. Sama seperti model 320x320, model ini mampu memprediksi seluruh SKU pada rak pertama dan kedua. Namun, model ini tidak berhasil memprediksi satupun objek yang ada pada rak ketiga. Pada rak keempat, model ini hanya mampu memprediksi satu dari 12 objek yang ada.

Gambar 4. 20 Hasil dari inferensi Model TFLite 320x320 di TensorFlow 2

Gambar 4. 21 Hasil dari inferensi Model TFLite 640x640 di TensorFlow 2

Hasil dari inferensi Model TFLite pada SSD MobileNetV2 FPNLite 1024x1024 di TensorFlow 2 dapat dilihat pada Gambar 4. 22. Masih sama seperti dua model TensorFlow 2 yang lain, model ini mampu memprediksi seluruh SKU pada rak pertama dan kedua. Pada rak ketiga, model ini hanya mampu memprediksi 8 dari 11 objek yang ada. Pada rak keempat, model ini tidak mampu memprediksi satupun objek yang ada.

Gambar 4. 22 Hasil dari inferensi Model TFLite 1024x1024 di TensorFlow 2

Perbedaan hasil prediksi dari setiap model dipengaruhi oleh berbagai faktor, seperti perbedaan arsitektur, perbedaan resolusi input, dan perbedaan versi framework. Arsitektur dari MobileNetV3 memiliki layer squeeze-and-excitation yang meningkatkan sensitivitas terhadap fitur penting dalam gambar. Perbedaan resolusi input juga memperngaruhi hasil prediksi karena resolusi yang lebih besar umumnya memberikan infomasi yang lebih mendetail yang membuat model memahami fitur yang lebih lengkap. Terakhir, penggunaan perbedaan versi framework juga mempengaruhi hasil pelatihan di mana TensorFlow 2 menggunakan eager-execution yang memberikan perbedaan bagaimana gradien dihitung dalam pelatihan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, beberapa kesimpulan dapat diambil sebagai berikut:

a. Peneliti berhasil mengimplementasikan deteksi objek untuk produk retail dengan perangkat bergerak. Pengembangan model memanfaatkan kerangka kerja TensorFlow 1 dengan arsitektur SSDMobileNetV2 dan TensorFlow 2 dengan arisitektur SSD MobileNetV2 FPNLite.
b. Terdapat empat variasi model TensorFlow yang berhasil dilatih, yakni TensorFlow 2 dengan model 320x320, model 640x640, model 1024x1024, dan TensorFlow 1 dengan model 1024x1024.
c. Skor mAP seluruh model asli TensorFlow 2 lebih unggul dibandingan skor mAP model TensorFlow 1. Namun, skor mAP pada model TFLite tidak jauh berbeda antara satu model dengan model lainnya.
d. Skor mAP TFLite yang didapatkan oleh keempat model tidak jauh berbeda. Terlihat dari perbedaan mAP model skor terendah dengan skor tertinggi hanya sebesar 0.02. Untuk model TFLite, perbedaan model skor terendah dengan skor tertinggi hanya sebesar 0.025.
e. Waktu inferensi dari setiap model berbeda-beda, bergantung pada ukuran input dari setiap model. Semakin besar ukuran input model, semakin lama waktu inferensi yang dibutuhkan.
f. Waktu inferensi tercepat didapatkan oleh SSD MobileNetV2 FPNLite 320x320 dengan waktu inferensi di perangkat bergerak sebesar 132 ms, lebih cepat dibandingkan model 640x640 dengan 413 ms, model 1024x1024 dengan 1039 ms, dan model TensorFlow 1 dengan .
g. Untuk penggunaan deteksi objek secara real-time, model SSD MobileNetV2 FPNLite 320x320 paling cocok untuk digunakan karena memiliki waktu inferensi yang cepat dengan pengurangan mAP yang tidak signifikan.
h. Untuk penggunaan deteksi objek yang mengutamakan akurasi, model SSD MobileNetV2 FPNLite 1024x1024 paling cocok untuk digunakan karena memiliki skor mAP yang paling tinggi diantara ketiga model lainnya, walaupun dengan inference time yang lebih lama.

i. Untuk penggunaan deteksi objek yang mempertimbangkan keseimbangan antara akurasi dan inference time, SSD MobileNetV3 Large 1024x1024 di TensorFlow 1 masih relevan untuk digunakan karena memiliki skor mAP yang cukup tinggi dan inference time yang masih cukup cepat.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa rekomendasi perbaikan yang bisa diterapkan pada pengembangan selanjutnya. Rekomendasi tersebut meliputi:

a. Penggunaan hyperparameter yang berbeda untuk pelatihan, seperti jumlah step dan batch size.
b. Penggunaan dataset dengan kondisi gambar dan distribusi kelas yang berbeda.
c. Penggunaan teknologi cloud computing untuk memanfaatkan aristektur deteksi objek yang lebih canggih namun tidak membebani komputasi perangkat bergerak karena proses deteksi dilakukan di cloud.

DAFTAR PUSTAKA

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., . . . Tucker, P. (2016). TensorFlow: A System for Large-Scale Machine Learning. *12th USENIX Symposium on Operating Systems Design*, 265-283.

Anica-Popa, I., Anica-Popa, L., Rădulescu, C., & Vrîncianu, M. (2020). THE INTEGRATION OF ARTIFICIAL INTELLIGENCE IN RETAIL: BENEFITS, CHALLENGES AND A DEDICATED CONCEPTUAL FRAMEWORK. *Artificial Intelligence in Wholesale and Retail*, 120-136. doi:10.24818/EA/2021/56/120

Ardiansyah, M. N., Muttaqin, P. S., Prasetio, M. D., & Novitasari, N. (2021). Object/Product Identification for Stock Taking Activities using Object Recognition . *JURNAL REKAYASA SISTEM DAN INDUSTRI*, 29-34.

Cai, Y., Wen, L., Zhang, L., Du, D., & Wang, W. (2021). Rethinking Object Detection in Retail Stores. *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 947-954.

Chong, T., Bustan, I., & Wee, M. (2016). Deep Learning Approach to Planogram Compliance.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: a Large-Scale Hierarchical Image Database. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 248-255). Miami: IEEE Computer Society.

Fighera, G. (2019). *Automatic Product Recognition using Deep Learning: a Retail Industry Application.* Padova: Università degli studi di Padova.

Forsyth, D., & Ponce, J. (2012). *Computer Vision: A Modern Approach.* Pearson.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* Cambridge: The MIT Press.

Google Research. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

Gruen, T. W., Corsten, D. S., & Bharadwaj, S. (2002). *Retail Out-of-Stocks: A Worldwide Examination of Extent, Causes and Consumer Responses.* Grocery Manufacturers of America, The Food Marketing Institute and CIES – The Food Business Forum.

Hausruckinger, G. (2006). Approaches to measuring on-shelf availability at the point of sale. *Roland Berger Strategy Consultants and ECR Europe*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition*, (pp. 770-778). Las Vegas.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv.1704.04861*.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., . . . Adam, H. (2019). Searching for MobileNetV3. *IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 1314-1324). Seoul: IEEE Computer Society.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., . . . Research, G. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3296-3297.

Jordan, M. I., & Mitchell, T. M. (2015). Machine Learning: Trends, Perspectives, and Prospects. *Science 349*, 255-260.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature volume 521*, 436–444.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 936-944.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Springer, Cham*, 21-37.

Manning, C. (2020). *Artificial Intelligence Definitions.* Stanford University Human-Centered Artificial Intelligence.

McCarthy, J. (2007). *What is Artificial Intelligence?* Stanford: Stanford University.

Meng, J., Jiang, P., Wang, J., & Wang, K. (2021). A MobileNet-SSD Model with FPN for Waste Detection. *Journal of Electrical Engineering & Technology*.

Moorthy, R., Behera, S., & SauravVerma. (2015). On-Shelf Availability in Retailing. *International Journal of Computer Applications*.

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1137-1149.

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach.* New Jersey: Pearson Education, Inc.

Sabina, Aneesa, & Haseena. (2022). Object Detection using YOLO And Mobilenet SSD: A Comparative Study. *International Journal of Engineering Research & Technology*, 134-138.

Sanchez, S. A., Romero, H., & Morales, y. (2021). A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework. *Expotecnología 2019 "Research, Innovation and Development in Engineering"*, 1.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4510-4520). Salt Lake City: IEEE Computer Society.

Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. *Neural Networks*, 85-117.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*.

Singh, A. N. (2020, April 25). *Image Recognition and Object Detection in Retail*. Retrieved from Ankit's Medium Web Site: https://ankitnsingh.medium.com/

Spielmaker, K. J. (2012). On-Shelf Availability in Retailing: A Literature Review and Conceptual Framework.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* London: Springer.

Yilmazer, R., & Birant, D. (2021). Shelf Auditing Based on Image Classification Using Semi-Supervised Deep Learning to Increase On-Shelf Availability in Grocery Stores. *sensors*.

LAMPIRAN

Hasil Evaluasi Keempat Model Penelitian untuk Setiap SKU

ID	Class	Objects	TF2 320x320	TF2 640x640	TF2 1024x1024	TF1 1024x1024
1	CARNATION Coffee-mate 495g	96	0.001254	0.001349	0.001329	0.742805
2	MILO ACTIVGO RTD 240ml	50	0.663697	0.692476	0.703266	0.742549
3	MILO ACTIVGO RTD HiCal 240ml	40	0.697077	0.802065	0.797403	1
4	NESCAFE Eclair Latte PET 220ml	25	0.780094	0.64046	0.729187	0.6791
5	NESCAFE Honey Latte PET 220ml	15	0.917725	0.793838	0.80407	0.584948
6	NESCAFE ChocoBananaLatte PET 220ml	14	0.526024	0.696219	0.678335	0.88
7	BEAR BRAND RTD Milk Tin 189ml	85	0.531684	0.58393	0.617544	0.455612
8	BEAR BRAND White Tea RTD Tin 140ml	63	0.733076	0.701018	0.73425	0.715421
9	BEAR BRAND White Malt RTD 140ml	80	0.683756	0.806719	0.92299	0.744809
10	DANCOW StrawberryFortigroUHT 110ml	29	0.780184	0.846112	0.897433	0.739902
11	DANCOW Coklat Fortigro UHT 110ml	45	0.503603	0.742179	0.762446	0.476484
12	CARNATION SCC 370g	125	0.311503	0.446832	0.351294	0.319231
13	NESCAFE Original Can 240ml	31	0.75459	0.81665	0.832511	0.583767
14	NESCAFE Latte Can 240ml	28	0.806314	0.795524	0.853073	0.527867
15	NESCAFE Mocha Can 240ml	27	0.755644	0.714995	0.797723	0.488523
16	MILO ACTIV-GO Nutriup 225ml	25	0.737119	0.838998	0.876927	0.676741
17	BEAR BRAND RTD Milk Tin (5(6x189ml)	17	0.71539	0.71848	0.79459	0.705882
18	DANCOW Strw Fortgr UHT 180ml	22	0.719381	0.803689	0.730709	0.465519
19	DANCOW Coklat Fortgr UHT 180ml	26	0.418218	0.523127	0.696869	0.286618
20	CARNATION Evaporated 405g	53	0.642739	0.652689	0.686592	0.49275
21	NESTLE GOODNES Kurma Milk UHT 180ml	25	0.836794	0.96	0.953103	0.913429
22	NESTLE GOODNES Plain Milk UHT 180ml	30	0.830892	0.889744	0.866667	0.669327
23	FRISIAN FLAG	265	0.231618	0.273038	0.290663	0.312574

24	ULTRA	148	0.059479	0.070671	0.130034	0.337728
25	NESTLE NONA Plain SBC Can 370g	33	0.974155	0.99167	0.999109	0.95441
26	NESTLE GOODNES Kurma Milk 189ml	11	0.839915	0.986014	0.815462	0.763373
27	NESTLE GOODNES Kencur Milk UHT 180m	47	0.557879	0.526945	0.484085	0.463744
28	NEST GOODNES JaheMaduMilk UHT 180ml	37	0.504194	0.499324	0.486486	0.480886
29	MILO ACTIV-GO Nutriup 225ml	6	1	1	1	0.958333
30	NESCAFE Coffee Cream UHT 180ml	29	0.863544	0.939509	0.910964	0.882881
31	NESCAFE Black UHT 180ml	34	0.712442	0.876587	0.872899	0.778309
32	INDOMILK	313	0.161231	0.389562	0.41445	0.154026
33	CAP ENAK	72	0.071079	0.103834	0.12155	0.14897
34	NESCAFE Kurma Latte PET 220ml	18	0.507856	0.457731	0.524433	0.304374
35	STARBUCKS Doubleshot Esp Latte 220ml	7	0.982143	0.982143	1	0.76024
36	STARBUCKS Doubleshot Mocha 220ml	6	1	1	1	1
37	OMELA	30	0.266667	0.287214	0.295652	0.342424
38	NESCAFE Cappuccino 220ml	39	0.948043	0.948043	0.95467	0.888953
39	NESCAFE Latte 220ml	38	0.923099	0.92743	0.933878	0.851531
40	NESCAFE Caramel Macchiato 220ml	16	0.875	0.870833	0.875	0.630221
41	NESCAFE French Vanilla UHT 180ml	18	0.827251	0.859645	0.759841	0.744623
42	GOOD DAY	47	0.532653	0.526966	0.667404	0.282496
43	GOLDA COFFEE	19	0.554982	0.573056	0.574012	0.464115
44	POKKA	8	0.781933	0.930556	0.906469	0.916667
45	MILO ACTIV-GO UHT Cmbk 180ml	52	0.688252	0.685404	0.744372	0.569385
46	DEL MONTE	35	0.900974	0.914286	0.978637	0.612976
47	MILO ACTIV-GO UHT Cmbk 9(4x180ml)	14	0.82154	0.808957	0.748129	0.575967
48	HILO	79	0.837472	0.942789	0.918971	0.509755
49	DANCOW Coklat Fortigro UHT 110ml	29	0.875714	0.890218	0.833088	0.59098
50	MILO ACTIV-GO UHT Cmbk 110ml	33	0.386789	0.432109	0.390166	0.741224
51	MILO ACTIV-GO UHT Cmbk 9(110ml)	19	0.852914	0.795479	0.738581	0.366381
52	TUJUH KURMA	26	0.801839	0.853855	0.856838	0.579921
53	DANCOW Strw Fortgr UHT 180ml	35	0.853494	0.870527	0.796083	0.700686

54	DANCOW Coklat Fortgr UHT 180ml	12	0.940705	0.947619	0.933862	0.776844
55	DANCOW StrawberryFortigroUHT 110ml	14	0.990476	1	1	0.967033
56	DANCOW Coklat Fortigro UHT 9(110ml)	14	1	0.975765	0.966882	1
57	DANCOW Vanila Fortigro UHT 9(110ml)	6	1	1	1	1
58	DANCOW StrawberryFortigroUHT9(110ml)	4	1	1	1	1
59	NESCAFE Thai Milk Coffee 220ml	6	1	1	1	1
60	STARBUCKS Frappuccino Coffee 280ml	4	1	1	1	1
61	BEAR BRAND GOLDY RTD MILKTIN 189	28	1	1	1	0.952381
62	NESCAFE Latte Can Cmbk 12(176+64ml)	2	1	1	1	1
63	GREENFIELDS	70	0.253481	0.515814	0.547962	0.111761
64	L-MEN	19	0.202193	0.43114	0.350253	0
65	TIGA SAPI	12	0.208333	0.216667	0.166667	0.166667
66	CARNATION SBC Plain Tube 6(180g)	4	1	1	1	1
67	MILO ACTIVGO RTD 240ml	9	0.883191	0.905556	0.988889	0.815037
68	MILO ACTIV-GO UHT Cmbk 6(110ml)	6	1	1	1	0.660063
69	MILO ACTIV-GO UHT Cmbk 9(180ml)	3	1	1	1	1
70	DANCOW Vanila Fortigro UHT6(110ml)	28	0.967078	0.995074	0.990148	0.983639
71	DANCOW Coklat FortigroUHT 6(110ml)	25	0.973037	0.990769	0.993846	0.956923
72	DANCOW Strw Fortigro UHT 6(110ml)	24	0.996795	1	1	0.993333
73	NESCAFE Ice Black 220ml	16	1	1	0.989149	0.990132
74	MILO ACTIV-GO Nutri Up RTD 220ml	763	0.981264	0.999067	0.997853	0.965411
75	NESCAFE Bis Koffee 220ml	288	0.995085	0.998344	0.997623	0.962766
76	NESTLE GOODNES KURMA AJWA MILK 189ML	227	1	1	1	1
77	MILO UHT Nutriactiv Cereal 180ml	711	0.957434	0.991447	0.993712	0.927966
78	MILO UHT Nutriactiv Banana 180ml	366	0.940093	0.992485	0.993208	0.941065
79	NESCAFE Ice Black 220ml	182	0.997995	1	1	0.984464
80	NESCAFE Ice Black Lychee 220ml	248	0.994888	0.997619	0.998573	0.980973
81	NESCAFE Latte 220ml	164	0.99931	1	1	0.995685

82	NESCAFE Caramel Macchiato 220ml	154	0.983937	0.996519	1	0.963608
83	NESCAFE Cappuccino 220ml	214	0.988318	0.999604	1	0.988541
84	DANCOW Vanila Fortigro UHT 110ml	153	0.973282	1	1	0.972896
85	DANCOW StrawberryFortigroUHT 110ml	80	0.905497	1	1	0.971609
86	DANCOW Coklat Fortigro UHT 110ml	125	0.933538	1	1	0.979168
87	DANCOW Vanila Fortigro UHT 110ml	10	0.966923	1	1	0.897565
88	DANCOW Plain Fortigro UHT 110ml	4	1	0.95	1	0.458333
89	DANCOW Coklat Fortigro UHT 9(110ml)	2	1	1	1	1
90	BOOST OPTIMUM ACB011 400g	3	1	1	1	1
91	BOOST OPTIMUM ACB011 800g	2	1	1	1	1
92	NUTREN DIABETES ACD009 400g	2	1	1	1	1
93	NUTREN DIAB ACD009 400g	2	1	1	1	1
94	NUTREN JNR PREBIO 1 ACB003 400g	2	1	1	1	1
95	NUTREN JUNIOR ACB003 800g	2	1	1	1	1
96	PEPTAMEN JUNIOR ACE002-2 400g	2	1	1	1	1
97	PEPTAMEN ACE003 400g	2	1	1	1	1
98	ISOCAL BIB 400g	2	1	1	1	1
99	NARAYA	10	0.046946	0.065455	0.193381	0.054798
100	WRP	27	0.614892	0.740741	0.505718	0