UNIVERSITAS INDONESIA

RANCANG BANGUN METODE *OBJECT DETECTION* YOLOV5 PADA RASPBERRY PI 4B UNTUK *SMART TROLLEY*

SKRIPSI

Muhammad As'ad Muyassir

1806199953

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

JUNI 2022

UNIVERSITAS INDONESIA

RANCANG BANGUN METODE *OBJECT DETECTION* YOLOV5 PADA RASPBERRY PI 4B UNTUK *SMART TROLLEY*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

Muhammad As'ad Muyassir

1806199953

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK KOMPUTER

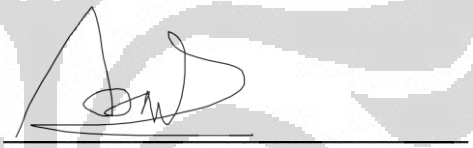
DEPOK

JUNI 2022

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama : **Muhammad As'ad Muyassir**
NPM : **1806199953**

Tanda Tangan : 
Tanggal : **23 Juni 2022**

LEMBAR PENGESAHAN

Skripsi ini diajukan oleh:
Nama : Muhammad As'ad Muyassir
NPM : 1806199953
Program Studi : Teknik Komputer
Judul Skripsi : Rancang Bangun Metode *Object Detection* YOLOv5 pada Raspberry Pi 4B untuk *Smart Trolley*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana mata kuliah Skripsi pada Program Studi Teknik Komputer Fakultas Teknik Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. Prima Dewi Purnamasari, S.T., M.T., M.Sc (……)

Penguji : I Gde Dharma Nugraha, S.T., M.T., Ph.D (……)

Penguji : Yan Maraden, S.T., M.T., M.Sc. (……)

Ditetapkan di : Fakultas Teknik
Tanggal : 14 Juli 2022

KATA PENGANTAR

Puji dan syukur dipanjatkan ke hadirat Allah SWT atas segala limpahan nikmat, rahmat, dan karunia-Nya, laporan skripsi dengan judul "Rancang Bangun Metode *Object Detection* YOLOv5 pada Raspberry Pi 4B untuk *Smart Trolley*" dapat diselesaikan dengan baik.

Laporan skripsi ini disusun untuk memenuhi salah satu syarat untuk dapat menyelesaikan program studi S1 pada jurusan Teknik Komputer Universitas Indonesia.

Dalam proses pembuatan laporan skripsi ini tentu tidak terlepas dari bantuan berbagai pihak lain baik secara langsung maupun tidak. Oleh sebab itu ucapan terima kasih dan apresiasi sebesar-besarnya diucapkan kepada pihak yang telah membantu, diantaranya adalah:

1) Dr. Prima Dewi Purnamasari S.T., M.Sc. selaku dosen pembimbing yang telah memberikan bantuan, arahan, masukan, serta saran-saran dalam proses penulisan skripsi.
2) Kedua Orang Tua yang selalu memberikan dukungan dan doa selama proses penulisan laporan skripsi.
3) Dr. Eng. Mia Rizkinia S.T., M.T. selaku dosen pembimbing akademis yang telah membantu dalam memberikan bantuan dan informasi akademis selama masa perkuliahan.
4) Segenap dosen Departemen Teknik Elektro Universitas Indonesia yang selalu memberikan ilmu yang bermanfaat selama masa perkuliahan.
5) Teman-teman program studi S1 Departemen Teknik Elektro, khususnya jurusan Teknik Komputer angkatan 2018 yang telah memberikan bantuan dan dukungan dalam melakukan penulisan laporan skripsi.

Penulis menyadari kekurangan laporan skripsi yang telah dibuat ini masih jauh dari kata sempurna, oleh karena itu penulis sangat terbuka terhadap segala kritik dan saran yang membangun agar penulis bisa menjadi lebih baik lagi. Semoga laporan

skripsi ini dapat bermanfaat bagi seluruh pembaca dan bagi para akademisi untuk perkembangan bidang keilmuan ke depannya.

Depok, Juni 2022

Muhammad As'ad Muyassir

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertandatangan di bawah ini:

Nama	: Muhammad As'ad Muyassir
NPM	: 1806199953
Program Studi	: Teknik Komputer
Fakultas	: Fakultas Teknik
Jenis Karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (***Non-exclusive Royalty-Free Right***) atas karya ilmiah saya yang berjudul:

RANCANG BANGUN METODE *OBJECT DETECTION* YOLOV5 PADA RASPBERRY PI 4B UNTUK *SMART TROLLEY*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 23 Juni 2022
Yang menyatakan

Muhammad As'ad Muyassir

ABSTRAK

Nama : Muhammad As'ad Muyassir
Program Studi : Teknik Komputer
Judul : Rancang Bangun Metode *Object Detection* YOLOv5 pada Raspberry Pi 4B untuk *Smart Trolley*

Supermarket merupakan tempat pilihan terbaik untuk berbelanja kebutuhan rumah saat ini karena pelanggan dapat memilih produk yang ingin dibelinya tanpa perlu mengantre. Namun untuk melakukan pembayaran saat ini pelanggan masih perlu mengantre di kasir. Oleh karena itu, penelitian ini akan mengimplementasikan sistem *cashierless* yang dapat melakukan *checkout* secara otomatis dan efisien sehingga pelanggan tidak perlu mengantre lagi di kasir. Sistem *cashierless* yang digunakan pada penelitian ini adalah *smart trolley*, sistem ini dapat melakukan deteksi produk yang masuk atau keluar dari troli pelanggan lalu melakukan *checkout* secara otomatis saat pelanggan keluar dari supermarket. Untuk dapat melakukan deteksi produk diperlukan model *machine learning* yang berjenis *object detection*. Model juga harus dapat diimplementasikan pada *edge device* karena deteksi akan dilakukan di troli yang memiliki keterbatasan ruang. Maka model yang digunakan adalah YOLOv5 karena memiliki akurasi serta performa tinggi supaya tetap dapat diimplementasikan pada *edge device*. Hasil pengujian variasi *backbone* menunjukkan *backbone original* lebih baik dari *backbone* Swin Transformer dengan nilai F1-Score sebesar 98.64%, ukuran model sebesar 7.7 MB, dan dapat berjalan dengan 3.87 FPS di komputer pengujian dan 0.74 FPS di Raspberry Pi 4B. Hasil pengujian variasi *dataset* menunjukkan kombinasi *dataset* bergerak dengan statis *blur* dapat menghasilkan model yang memiliki akurasi yang paling baik dengan nilai 99.53% pada fase pelatihan dan 99.44% pada fase testing. Hasil pengujian intensitas cahaya menunjukkan penggunaan lampu untuk meningkatkan pencahayaan di sekitar wilayah deteksi di dalam troli dapat meningkatkan F1-Score hasil deteksi yang dilakukan hingga 63.55%. Hasil pengujian variasi kecepatan produk menunjukkan kecepatan ideal yang dapat digunakan pada saat proses deteksi di komputer pengujian adalah hingga 36 cm/s dan untuk proses yang dilakukan di Raspberry Pi 4B adalah di bawah 7 cm/s. Hasil pengujian dengan penambahan *sampling rate* dapat mendeteksi produk di komputer pengujian dengan kecepatan hingga 124 cm/s pada produk-produk dengan ukuran yang cukup lebar.

Kata Kunci:

Cashierless; Smart Trolley; Machine Learning; Object Detection; Edge Device; YOLOv5; Swin-Transformer;

ABSTRACT

Name : Muhammad As'ad Muyassir
Study Program : Computer Engineering
Title : Development of YOLOv5 Object Detection on Raspberry Pi 4B for Smart Trolley

Supermarkets are the best place to shop for home needs today because customers can choose what products they want to buy without the need to queue. However, today customers still need to queue at the cashier to make payments. Therefore, this research will implement a cashier-less system that can do checkout automatically and efficiently so that customers don't have to queue at the cashier anymore. The cashier-less system used in this study is a smart trolley, this system can detect products entering or leaving the customer's trolley and then checkout automatically when the customer leaves the supermarket. To be able to perform product detection, a machine learning model of the object detection type is needed. The model must be able implemented on edge devices because the detection will be done in the cart with limited space. So, the model used is YOLOv5 because it has high accuracy and performance so it can implement on edge devices. The backbone variation test results show that the original backbone is better than the Swin-Transformer backbone with an F1-Score value of 98.64%, a model size of 7.7 MB, and can run with 3.87 FPS on a test computer and 0.74 FPS on a Raspberry Pi 4B. The dataset variation test results show that the combination of moving datasets with static blur can produce a model with the best accuracy of 99.53% in the training phase and 99.44% in the testing phase. The light intensity variation test results show that the use of lamps to increase the lighting around the detection area in the trolley can increase the F1-Score of the detection results made up to 63.55%. The product speed variation results show that the ideal speed that can use during the detection process on the testing computer is up to 36 cm/s and for the process carried out on the Raspberry Pi 4B it is below 7 cm/s. The sampling rate addition results can detect products on the test computer at speeds up to 124 cm/s on products with a wide size.

Key words:

Cashierless; Smart Trolley; Machine Learning; Object Detection; Edge Device; YOLOv5; Swin-Transformer;

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Toko swalayan adalah sebuah tempat yang menyediakan berbagai produk yang sering dikonsumsi sehari-hari atau produk-produk tersebut bisa disebut juga sebagai *Fast-Moving Consumer Good* (FMCG) [1]. Sampai saat ini, toko swalayan masih menjadi tujuan utama dalam mencari kebutuhan sehari-hari karena dinilai lebih mudah, cepat, dan murah dibandingkan dengan toko kelontong. Toko swalayan juga disenangi karena pelanggan dapat dengan leluasa memilih sendiri barang yang diinginkannya lalu membayarnya ke kasir tanpa harus mengantre untuk melakukan pemesanan. Pembayaran yang ditawarkan oleh toko swalayan juga mempermudah pelanggan dengan mendukung berbagai macam sistem pembayaran baik itu tunai ataupun non-tunai [2].

Saat ini, bisnis E-commerce sedang berkembang dengan sangat signifikan karena menawarkan sistem berbelanja yang baru dengan berbagai kemudahan yang ada di dalamnya. Namun, hal tersebut masih belum bisa menggeser kebiasaan banyak orang yang lebih memilih berbelanja di toko swalayan terutama untuk berbelanja kebutuhan sehari-hari. Hal tersebut terjadi karena pada E-commerce terdapat biaya tambahan untuk ongkos pengiriman serta perlu waktu untuk menunggu proses pengiriman dilakukan. Terlebih lagi toko swalayan berjenis minimarket sudah mudah untuk dijumpai dan menawarkan berbagai produk pokok sehari-hari [3].

Walaupun toko swalayan masih diminati, namun swalayan tetap harus mengikuti perkembangan zaman terutama dalam bidang teknologi. Kemajuan pesat dalam bidang teknologi akan mengubah perilaku pelanggan, sehingga swalayan harus mengikutinya untuk dapat bersaing dengan bisnis serupa. Teknologi juga dapat membawa banyak manfaat terutama dalam hal efisiensi dan peningkatan kepuasan pelanggan. Apalagi pelanggan dengan usia muda sudah terbiasa dan lebih suka menggunakan teknologi walaupun masih baru untuk mempermudah dirinya [4].

Inovasi teknologi pada sistem *checkout* adalah salah satu yang terpenting, karena sistem *checkout* menggunakan kasir akan membuat sebuah antrean ketika banyak orang yang ingin membayar pada saat yang bersamaan. Padahal, toko swalayan telah mempermudah pelanggan karena dapat memilih produk sendiri tanpa harus memesan terlebih dahulu. Sebagian besar pelanggan tidak suka mengantre karena lebih peduli pada seberapa panjang antrean dibandingkan dengan seberapa cepat antrean bergerak [5]. Antrean di kasir juga harus dikurangi karena akhir dari suatu pengalaman adalah hal yang paling diingat oleh pelanggan [6]. Tanpa inovasi, toko swalayan harus memikirkan seberapa panjang antrean yang mungkin terjadi dengan tetap mempertahankan jumlah kasir yang efisien.

Sistem *checkout* konvensional telah dikembangkan menjadi *Self-checkouts* (SCOs) atau *self-service checkouts* dengan memanfaatkan berbagai macam teknologi. SCOs dinilai lebih efisien jika diimplementasikan dengan tepat karena dapat meningkatkan pengalaman pelanggan serta mengurangi biaya hingga 81% [6]. SCOs dapat mengurangi biaya karena hanya memerlukan biaya implementasi di awal dengan perawatan yang murah. Sistem SCOs yang paling sederhana adalah dengan membuat pelanggan melakukan *scan barcode* sendiri untuk setiap produk yang dibeli. Namun, sistem tersebut mengharuskan pelanggan untuk melakukan proses *scanning* yang cukup sulit karena harus melakukan *scan* tepat pada *barcode* yang ada di produk dan rentan kesalahan sehingga membuatnya kurang efisien [4]. Lalu sistem SCOs dikembangkan menjadi *cashierless system* dengan memanfaatkan teknologi *embedded system* yang menggunakan sensor *Radio Frequency Identification* (RFID) untuk mendeteksi barang yang dibeli. Namun, saat melakukan *scanning* menggunakan sensor RFID terdapat kemungkinan beberapa produk tidak terdeteksi secara bersamaan karena terjadinya *collision* pada saat proses deteksi dilakukan. RFID juga memerlukan Tag RFID yang membuat harga jual produk menjadi naik [7]. *Cashierless system* yang saat ini banyak dikembangkan telah memanfaatkan *Machine Learning* (ML) untuk dapat mendeteksi produk yang ingin dibeli oleh pelanggan hanya berdasarkan informasi produk saja. Namun, jika sistem deteksi diletakkan pada lokasi yang tidak tepat serta menggunakan model yang dilatih dengan kurang baik, maka deteksi produk akan sulit untuk dilakukan. Untuk melakukan deteksi produk dengan baik sekaligus

mempermudah pelanggan maka *cashierless system* diletakkan pada keranjang belanja. Keranjang belanja yang memiliki implementasi teknologi di dalamnya untuk mempermudah proses belanja biasa disebut sebagai *smart trolley* [8].

Oleh karena itu, diperlukan penelitian untuk mengimplementasikan *cashierless system* berbasis *Object Detection* menggunakan *Embedded System* yang diletakkan pada keranjang belanja atau biasa disebut *smart trolley*. Sistem dibuat terintegrasi dengan keranjang belanja karena diharapkan dapat memberi pengalaman berbelanja tanpa hambatan. Dengan adanya penelitian ini diharapkan dapat memberi gambaran sistem berbelanja menggunakan *cashierless system* yang mudah untuk digunakan dan dapat mendeteksi produk dengan akurat.

1.2. Rumusan Masalah

Berdasarkan permasalahan yang telah diuraikan pada latar belakang, dapat diketahui bahwa sistem *checkout* saat ini masih kurang efisien. Maka rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana cara membuat *cashierless system* yang berbasis *smart trolley*?
2. Bagaimana cara merancang dan mengimplementasikan metode *Object Detection* untuk produk FMCG yang keluar dan masuk keranjang?
3. Bagaimana cara meningkatkan akurasi dan efisiensi *Object Detection* yang diimplementasikan pada *Embedded System*?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan *cashierless system* berbasis *smart trolley* untuk dapat mendeteksi beberapa jenis produk FMCG.
2. Menerapkan teknologi *Object Detection* pada Raspberry Pi 4B untuk dapat mendeteksi produk yang keluar dan masuk keranjang belanja.

1.4. Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Jenis produk yang dapat dideteksi hanya yang terdapat pada *dataset*.
2. Sistem diimplementasikan pada Raspberry Pi 4 Model B.
3. Kamera dan produk tidak banyak tertutup oleh apapun.
4. Produk harus terdeteksi saat sedang masuk atau keluar keranjang.
5. Hanya terdapat satu produk yang berada di depan kamera.

1.5. Metodologi Penelitian

Metode yang digunakan selama penelitian ini adalah sebagai berikut:

1. Studi Literatur

   Studi literatur dilakukan untuk mempelajari *cashierless system*, *Image Processing, Object Detection,* diagram arsitektur *Object Detection*, cara melatih model, serta bahan lainnya yang berkaitan dengan penelitian ini.
2. Mencari dan Membuat *Dataset*

   Mencari dan membuat *dataset* yang berisi gambar berbagai jenis produk FMCG serta mengolahnya dengan beberapa metode.
3. Membuat Rancangan Sistem *smart trolley*

   Membangun rancangan sistem *smart trolley* dan prototipe keranjang belanja serta menentukan model *Object Detection* yang akan digunakan.
4. Melatih Model

   Melatih model pada seluruh *dataset* yang sudah diproses lalu melakukan evaluasi dari hasil yang didapatkan.
5. Implementasi Sistem

   Melakukan implementasi model yang sudah dilatih ke dalam *Embedded Device* lalu memasangkannya ke dalam prototipe keranjang belanja.
6. Simulasi dan Pengujian

   Melakukan simulasi dan pengujian untuk memastikan sistem dapat berjalan sesuai dengan rancangan.
7. Melakukan Analisis

   Menganalisis hasil dari simulasi dan pengujian implementasi sistem.
8. Memberi Kesimpulan

   Mengambil kesimpulan berdasarkan hasil yang telah didapat.

1.6. Sistematika Penulisan

Sistematika Penulisan pada penelitian ini dibagi ke dalam 5 bab, yaitu:

BAB I PENDAHULUAN

Bab ini berisi pembahasan latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, dan metodologi penelitian ini.

BAB II LANDASAN TEORI

Bab ini berisi pembahasan seluruh teori yang diperlukan untuk menjadi dasar perancangan dan implementasi *cashierless system* berbasis *smart trolley* pada penelitian ini.

BAB III PERANCANGAN SISTEM

Bab ini berisi perancangan dari *cashierless system* berbasis *smart trolley* beserta model yang akan digunakan pada penelitian ini.

BAB IV IMPLENTASI DAN PEMBAHASAN

Bab ini berisi implementasi dan pembahasan dari hasil *cashierless system* berbasis *smart trolley*.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil penelitian dan saran untuk penelitian selanjutnya.

BAB 2
OBJECT DETECTION YOLOV5 PADA RASPBERRY PI 4B UNTUK *SMART TROLLEY*

2.1. *Cashierless System* berbasis *Smart Trolley*

Cashierless system adalah sistem yang dikembangkan untuk dapat menggantikan sistem *checkout* melalui kasir dengan sistem *checkout* yang otomatis. Sistem *cashierless* memanfaatkan berbagai macam teknologi yang mendukung seperti RFID, *embedded system*, *Machine Learning*, dll. Sistem *cashierless* diharapkan dapat meningkatkan pengalaman berbelanja bagi pelanggan dan meningkatkan efisiensi dalam manajemen toko [9].

Terdapat beberapa jenis teknologi yang ditawarkan oleh sistem *cashierless* diantaranya adalah *Mobile and Augmented Technology*, *Smart Trolley*, dan *automated store-level solutions using AI* [10]. Teknologi yang akan digunakan pada penelitian ini adalah sistem *cashierless* yang berjenis *Smart Trolley* [8]. Teknologi tersebut dipilih karena troli merupakan hal yang sudah biasa digunakan di toko swalayan, dapat menghasilkan sistem *cashierless* yang bebas hambatan, mudah untuk dipindahkan jika toko ingin berpindah lokasi, dan toko dapat menentukan berapa banyak troli yang akan digunakan sesuai kebutuhan dan anggaran. Pada Gambar 2.1 terdapat gambar contoh *smart trolley* yang saat ini telah diimplementasikan oleh Amazon untuk sistem perbelanjaannya.

Gambar 2.1 *Smart trolley* yang digunakan di Amazon

2.2. *Machine Learning*

Machine Learning (ML) adalah salah satu bagian dari *Artificial Intelligence* yang membangun model dengan belajar secara mandiri pada data sampel. Proses belajar yang dilakukan pada ML biasa dikenal sebagai data *training*, proses tersebut akan mempelajari pola yang terdapat di dalam data yang diberikan. ML dibuat supaya tidak perlu lagi melakukan penulisan fungsi matematis untuk melakukan prediksi atau pengambilan keputusan. Untuk membuat suatu sistem hanya perlu menyiapkan berbagai data yang menggambarkan *input* dan *output* dari suatu sistem lalu melatihnya ke dalam model ML dan ML akan memberikan hasil model yang dapat menjalankan tugas sesuai dengan data *input* dan *output* pada data *training* [11].

Pada ML terdapat tiga jenis yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning*. Metode *Supervised Learning* adalah metode yang akan digunakan pada penelitian kali ini. Pada *Supervised Learning* mesin akan mempelajari fungsi yang melakukan pemetaan input menjadi output berdasarkan contoh data yang diberikan dari *dataset* yang benar. Pada *Supervised Learning* diperlukan bantuan manusia untuk membuat *dataset* yang benar sesuai *input* dan *output* yang diharapkan. Mesin diperlukan untuk dapat mempelajari data yang sangat banyak pada ML karena terkadang banyak data yang terlalu kompleks sehingga tidak dapat diinterpretasikan oleh manusia. Pada ML mesin dapat mempelajari suatu fungsi sendiri tanpa harus dibuat secara manual oleh manusia melalui pendekatan matematika dalam mengolah *dataset*. Pada *Unsupervised Learning* metode pembelajaran yang dilakukan oleh mesin dalam menghasilkan sistem yang diinginkan tidak perlu lagi dibantu oleh manusia untuk memberikan *dataset* dengan pemetaan *input* dan *output* yang benar. *Unsupervised Learning* dirancang untuk dapat menemukan dan memberikan *insight* menarik dari suatu data yang diberikan. *Unsupervised Learning* biasa digunakan untuk kebutuhan *clustering* dan *feature reduction*. Pada *Reinforcement Learning* sistem pembelajaran yang dapat membuat model dapat mempelajari apakah yang diperbuat olehnya itu benar atau salah. Cara untuk mempelajari hal tersebut adalah dengan *feedback* yang diberikan oleh lingkungan yang berupa *reward* dan *punishment*. *Feedback* tersebut akan digunakan untuk sistem supaya dapat

mengetahui seluruh hal yang perlu dilakukan olehnya dan tidak boleh dilakukan olehnya. Model akan terus mempelajari hal baru saat dirinya diberikan *feedback* yang baru juga [12].

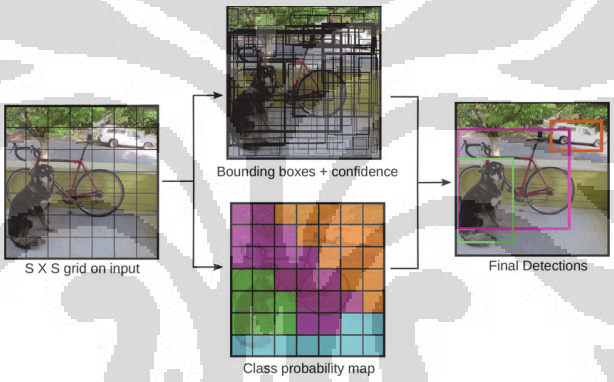
2.3. Object Detection

Object Detection adalah salah satu pengembangan dari *Computer Vision* yang dapat melakukan deteksi objek terhadap gambar digital yang diberikan. Sistem *Object Detection* merupakan pengembangan dari *Image Classification* karena dapat melakukan deteksi pada beberapa objek sekaligus serta mengetahui lokasi objek tersebut pada gambar. Sistem *Object Detection* menjadi salah satu dasar dari algoritma baru seperti *instance segmentation* dan *object tracking*. Sistem *Object Detection* terbagi menjadi dua jenis yaitu *general object detection* yang mendeteksi objek yang bersifat *general* dan *detection applications* yang mendeteksi objek yang lebih spesifik [13]. Salah satu model untuk melakukan deteksi objek yang terkenal adalah *You Only Look Once* (YOLO) karena dapat memberikan akurasi yang tinggi namun tidak menggunakan *resource* yang besar.

Algoritma *Object Detection* yang saat ini sedang banyak dikembangkan ke beberapa fokus diantaranya menciptakan *object detection* yang ringan untuk mempercepat proses deteksi. Keringanan dalam menjalankan model juga bertujuan supaya model dapat dijalankan pada perangkat yang bersifat *mobile* seperti *smartphone* dan *embedded*. Salah satu contoh model yang cukup mementingkan keringanan model serta kecepatan deteksi adalah model YOLOv5 yang akan dipakai pada penelitian kali ini. Selain keringanan fokus yang lainnya adalah untuk membuat *object detection* yang dispesifikasikan untuk menyelesaikan suatu masalah pada domain khusus. Karena kebutuhan *object detection* saat ini sudah sangat meluas dan dibutuhkan untuk implementasi pada suatu kebutuhan khusus untuk masalah di kehidupan nyata, sehingga diperlukan implementasi yang nyata juga untuk dapat memecahkan masalah tersebut. Pada penelitian ini juga akan membuat sistem dengan *object detection* pada domain khusus yaitu produk supermarket atau FMCG. Sistem yang dibuat juga harus dapat diimplementasikan pada keadaan nyata di supermarket [14].

2.4. You Only Look Once

You Only Look Once (YOLO) adalah salah satu model *Object Detection* yang menggunakan *regression* dalam mendeteksi objek seperti pada Gambar 2.2. Karena YOLO mendeteksi objek menggunakan sistem *regression*, maka YOLO dapat menjadi model yang sangat cepat dalam melakukan deteksi objek dengan penggunaan *resource* yang sedikit. YOLO juga mendukung deteksi objek secara *real-time* dengan *delay* kurang dari 25 detik. YOLO menggunakan gambar sebagai *input* model dan menggunakan posisi *bounding box* sebagai *output*-nya. YOLO menggunakan sistem *grid cell* seperti Gambar 2.2 yang digunakan untuk melakukan deteksi objek di setiap *cell* serta dapat memiliki kelasnya masing-masing yang membuat YOLO dapat mendeteksi beberapa objek dalam waktu yang cepat [15].

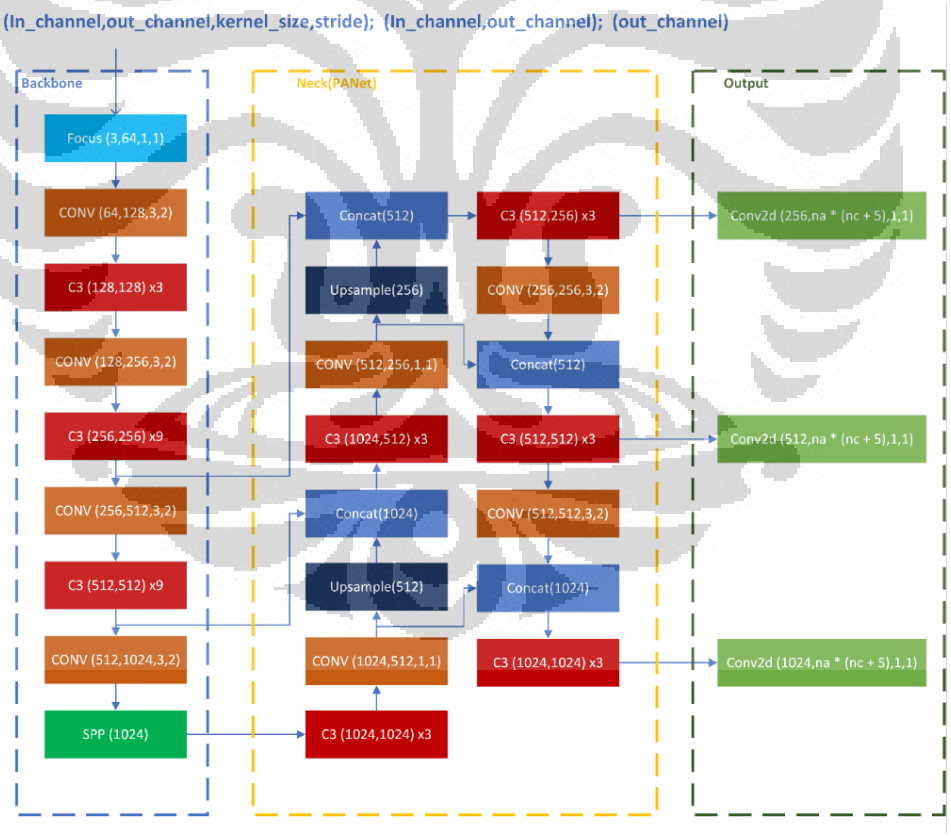


Gambar 2.2 *Regression* dan *grid cell* di YOLOv5 [16]

YOLO dibangun dengan menggunakan tiga komponen utama yaitu *Backbone* yang berisi *layer Convolutional Neural Network* (CNN) untuk menghasilkan fitur terperinci dari suatu gambar. Lalu komponen selanjutnya adalah *Neck* yang merupakan *layer* yang menggabungkan beberapa fitur dari gambar dan mengirimkan fitur gambar ke *prediction layer*. Komponen terakhir adalah *Head* yang merupakan *layer* yang dapat memprediksi fitur gambar, membuat *bounding box*, dan memprediksi kelas dari gambar [17].

YOLO sudah berkembang dengan sangat cepat, telah terdapat beberapa versi YOLO yang sudah dibuat. Pada penelitian ini digunakan YOLOv5 yang merupakan generasi YOLO terbaru yang ada saat penelitian ini dilakukan. YOLOv5

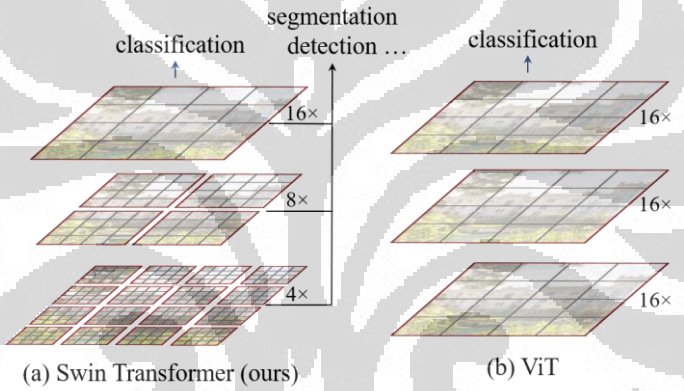
dikembangkan sama seperti YOLO sebelumnya yang mengutamakan kecepatan atau performa tanpa mengorbankan akurasi. YOLOv5 dapat memberikan tingkat akurasi yang lebih tinggi dari YOLOv4 namun dengan penggunaan *resource* yang lebih rendah. Pada Gambar 2.3 disajikan ilustrasi dari arsitektur pada model YOLOv5. Arsitektur tersebut akan digunakan sebagai referensi model YOLOv5 dan juga *backbone original* yang akan digunakan pada penelitian ini. YOLOv5 menggunakan arsitektur yang sama dengan YOLO sebelumnya dengan menerapkan susunan Backbone, Neck dengan PANet, dan output. Yang membedakan YOLOv5 adalah penggunaan layer yang lebih sederhana dibandingkan YOLO sebelumnya. Pada Gambar 2.3 juga dapat dilihat layer yang digunakan pada YOLOv5 hanya Convolution, C3, dan Concat saja sedangkan pada YOLOv4 layer yang digunakan adalah kombinasi dari Convolution dengan berbagai layer lainnya [16].



Gambar 2.3 Arsitektur YOLOv5 [16]

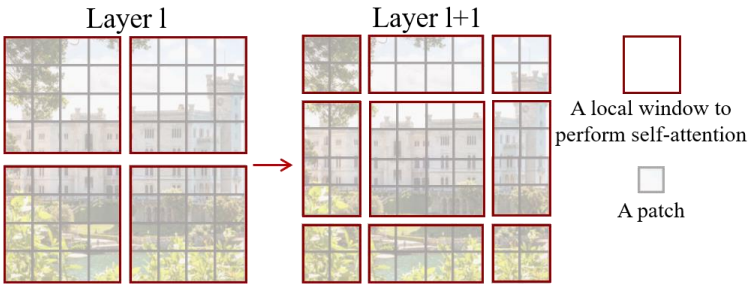
2.5. Swin Transformer

Swin Transformer adalah salah satu model yang dibuat untuk dapat digunakan sebagai *general-purpose backbone* untuk berbagai kebutuhan *computer vision*. *Swin transformer* terinspirasi dari implementasi arsitektur menggunakan *transformer* pada *Natural Language Processing* (NLP) yang dapat melakukan *self-attention* untuk mendapatkan hubungan dan konteks antara kata. Hal tersebut juga ingin diterapkan pada gambar untuk bisa melakukan *self-attention* yang dapat memberikan hubungan antar bagian pada suatu gambar. *Swin transformer* akan melakukan ekstraksi fitur dimulai dari gambar dibagi secara detail hingga umum seperti pada Gambar 2.4.

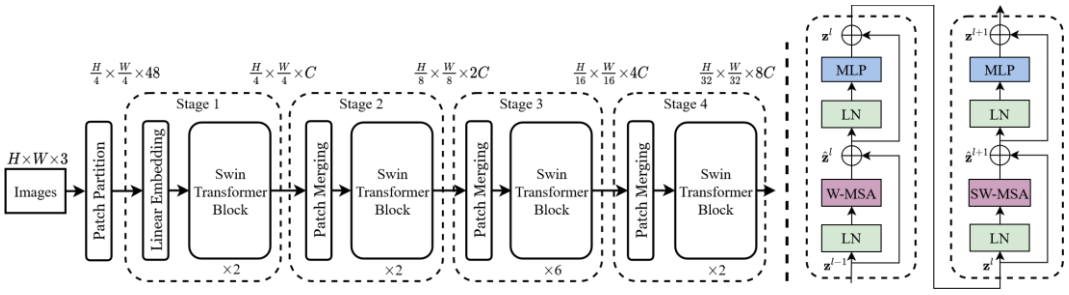


Gambar 2.4 Ekstraksi fitur pada *swin transformer* [18]

Swin transformer akan membagi gambar menjadi beberapa *window* yang digunakan untuk melakukan *self-attention* antar bagian atau *patch*. Kemudian *window* akan digeser seperti pada Gambar 2.5 untuk memberikan hubungan dari *window* sebelumnya.



Gambar 2.5 S*hifted windows* pada *swin transformer* [18]



Gambar 2.6 Arsitektur *swin transformer* [18]

Pada Gambar 2.6 terdapat arsitektur dari model *swin transformer* yang berasal dari model berjenis *tiny*. Pada model ini gambar akan dibagi ke dalam beberapa *patch* untuk dapat melakukan *self-attention* pada gambar secara detail. Kemudian pada tahap selanjutnya beberapa *patch* akan digabung untuk dilakukan *self-attention* pada gambar yang lebih *general* [18].

2.6. *Evaluation Metrics*

Evaluation Metrics adalah suatu nilai yang digunakan untuk memberikan gambaran performa dari model yang sedang atau sudah dilatih. Dengan *Evaluation Metrics* model dapat diukur seberapa baik, dapat diketahui jika terjadi *overfit,* dapat juga mengetahui permasalahan jumlah data atau persebaran data yang kurang baik. Pada penelitian kali ini digunakan beberapa jenis *Evaluation Metrics* yaitu *F1-Score, Precision, Recall,* serta *Mean Average Precision* (mAP) [19].

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP = True positive
TN = True negative
FP = False positive
FN = False negative

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$$\mathrm{AP} = \int_0^1 p(r)dr$$

Gambar 2.7 Cara mendapatkan nilai *Evaluation Metrics* [20], [21]

2.7. Roboflow

Roboflow adalah salah satu *tools* yang dapat digunakan untuk mengolah data gambar supaya sesuai dengan format yang dibutuhkan dan dapat digunakan pada proses pelatihan model ML. Roboflow memiliki fungsi utama yang dapat digunakan untuk membuat anotasi dan melakukan *augmentasi* dari *dataset*. Proses anotasi *dataset* dilakukan untuk memberi data posisi produk yang terdapat dalam gambar sebagai *output* yang diharapkan dapat diprediksi oleh model. Roboflow akan digunakan untuk melakukan anotasi pada penelitian ini untuk dapat mempermudah proses anotasi karena terdapat banyak fitur yang bisa dimanfaatkan.

Roboflow memiliki banyak fitur di dalamnya dan salah satu fiturnya adalah bersifat *online* sehingga dapat dengan mudah digunakan di mana saja dan kapan saja. Karena sifatnya yang *online* juga membuat Roboflow memiliki fitur untuk berkolaborasi dalam melakukan proses anotasi *dataset* sehingga dapat dilakukan secara bersama-sama dengan orang lain yang berada di lokasi yang berbeda. Roboflow juga dapat menghasilkan *dataset* dengan berbagai format anotasi yang popular dan juga dapat digunakan untuk mengubah dari suatu format anotasi ke format anotasi lainnya dengan mudah. Lalu hasil *dataset* tersebut dapat langsung diekspor menjadi *link zip dataset* yang bisa langsung diunduh dan digunakan melalui Google Colab, sehingga *dataset* yang telah diolah tidak memerlukan banyak ruang untuk menyimpannya di lokal dan harus mengunggahnya ke Google Colab setiap ingin digunakan. Untuk membuat variasi dari *dataset*, Roboflow telah menyediakan berbagai macam *augmentasi* yang dapat digunakan untuk menghasilkan *dataset* dengan berbagai macam variasi sesuai dengan kebutuhan. Namun untuk dapat menghasilkan banyak variasi *augmentasi* diperlukan *upgrade* akun dengan cara berlangganan, jika tidak berlangganan maka variasi *augmentasi* yang dapat dihasilkan hanya maksimal tiga kali dari jumlah awal *dataset* saja. Jika tidak berlangganan terdapat limitasi lain juga seperti limitasi jumlah total gambar yang dapat dianotasi atau diolah yaitu maksimal dengan jumlah 10000 gambar dan hanya bisa berkolaborasi dengan maksimal sepuluh pengguna lainnya saja [22].

Untuk melakukan anotasi, seluruh *dataset* harus diunggah terlebih dahulu ke Roboflow. Setelah *dataset* diunggah maka Roboflow akan mengecek apakah

dataset yang diunggah sudah terdapat anotasi di dalamnya atau belum, jika belum maka Roboflow akan meminta untuk menentukan siapa yang akan melakukan anotasi tersebut. Setelah ditetapkan maka orang tersebut dapat melakukan anotasi langsung pada Roboflow dengan menarik kotak pembatas dan menyesuaikannya pada objek yang ingin dianotasi. Setelah seluruh *dataset* dianotasi maka *dataset* sudah bisa diekspor dengan berbagai metode *split dataset* yang bisa dilakukan secara otomatis pada Roboflow. Metode *split dataset* juga dapat dilakukan secara manual jika tidak ingin data terbagi secara acak. Setelah dibagi, *dataset* dapat diproses untuk diubah ukuran, kontras, dan nama kelasnya serta dapat dilakukan berbagai *augmentasi* seperti menambah *blur* dan *noise* langsung pada Roboflow. Setelah itu *dataset* sudah siap diekspor ke berbagai format yang salah satunya adalah format YOLOv5. Format penulisan anotasi yang sesuai dengan ketentuan pada model YOLOv5, yaitu seperti pada Gambar 2.8 dengan format *file* txt. Satu gambar harus memiliki satu *file* anotasi yang berisi seluruh *bounding box* yang dipisahkan dengan baris baru seperti pada Gambar 2.9 [23].

Gambar 2.8 Format anotasi pada model YOLOv5

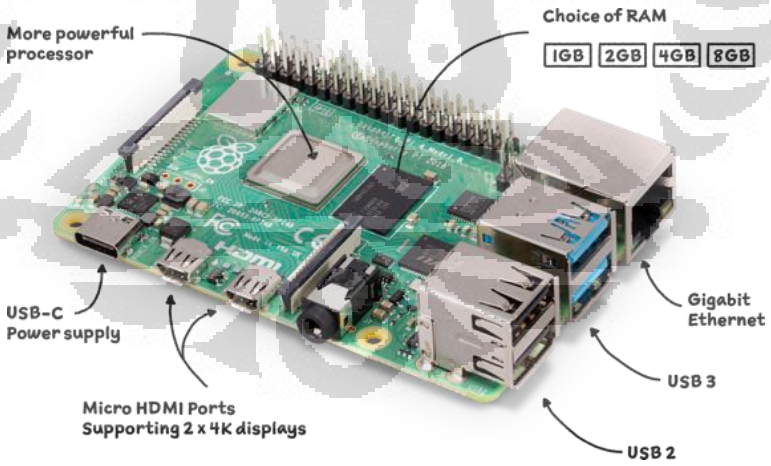
001.txt

```
1 0.617 0.3594420600858369 0.114 0.17381974248927037
1 0.094 0.38626609442060084 0.156 0.23605150214592274
```

Gambar 2.9 Contoh *file* anotasi untuk gambar yang bernama 001 [23]

2.8. Raspberry Pi 4B

Raspberry Pi 4B adalah perangkat *single-board computer* (SBC) yang dapat menjalankan berbagai hal yang dapat dilakukan oleh komputer biasa namun memiliki ukuran yang sama dengan kartu kredit. Raspberry Pi 4B dapat dihubungkan dengan perangkat komputer lainnya seperti *keyboard*, *mouse*, dan monitor [24]. Raspberry Pi 4B juga dapat dikoneksikan dengan dua monitor sekaligus dengan resolusi hingga 4K, memiliki performa tinggi dengan prosesor *quad-core* 64 bit, memiliki Bluetooth 5.0 serta Wireless LAN *dual-band*, terdapat USB 3.0 dengan kecepatan hingga 5 Gbit/s, dan memiliki pilihan RAM hingga 8 GB [25]. Pada Gambar 2.10 dapat dilihat bentuk Raspberry Pi 4B beserta komponen dan *port* yang dimilikinya.



Gambar 2.10 Komponen Raspberry Pi 4B [25]

Raspberry Pi 4B termasuk ke dalam perangkat *embedded* karena Raspberry Pi 4B dapat melakukan komunikasi dengan perangkat *embedded* lain, sensor, dan *actuator* melalui *pin* GPIO yang tersedia. Raspberry Pi 4B juga dapat digunakan sebagai perangkat *Internet of Things* (IoT) karena sudah dapat terhubung ke internet

melalui Wi-Fi ataupun Ethernet. Raspberry Pi 4B dipilih pada penelitian kali ini karena banyak dijual di Indonesia, mudah untuk dioperasikan, mudah untuk mengimplementasi program, dan memiliki harga yang lebih murah dibandingkan perangkat *embedded* lainnya seperti Intel Edison, Mediatek linkit one, NVIDIA Jetson Nano, dll [26]. Selain itu, jika menggunakan perangkat *embedded* dengan harga yang lebih murah dari Raspberry Pi 4B maka komputasi yang dibutuhkan untuk menjalankan model YOLOv5n dengan 4.5 GFLOPS tidak dapat terpenuhi [16], [27].

BAB 3
PERANCANGAN SISTEM

3.1. Rancangan Sistem

Penelitian ini dilakukan untuk menghasilkan sistem yang dapat mendeteksi produk FMCG menggunakan *Object Detection* yang diimplementasikan pada perangkat *Single Board Computer* (SBC) Raspberry Pi 4B. Sistem dapat menerima masukan yang berupa gambar melalui tangkapan kamera yang dihubungkan ke Raspberry Pi 4B. Kemudian sistem akan melakukan deteksi objek menggunakan model yang sudah dilatih pada *dataset* FMCG. Setelah mendeteksi, sistem akan mengetahui produk apa yang dimasukkan atau dikeluarkan dari keranjang belanja. Sistem ini diharapkan dapat menghilangkan antrean pada saat *checkout* karena sistem telah mengetahui produk apa saja yang ada di dalam keranjang belanja. Model yang akan digunakan dalam proses *Object Detection* pada penelitian ini adalah YOLOv5. Sistem *smart trolley* yang dibuat harus bisa mendeteksi produk yang bergerak secara *real-time* dengan model yang dapat dijalankan pada Raspberry Pi 4B.

Untuk dapat menghasilkan sistem yang sesuai, maka diperlukan beberapa kriteria yang harus dipenuhi sebagai berikut:

1. Sistem dapat mengenali produk yang terdapat pada *dataset* secara *real-time*.
2. Sistem dapat mendeteksi satu produk yang berada di depan kamera.
3. Sistem dapat mengetahui apakah produk sedang masuk ke keranjang atau keluar dari keranjang.
4. Model ML dapat dilatih menggunakan *custom dataset*.
5. Model ML dapat dikonversi menjadi jenis TensorFlow Lite.
6. Model ML dapat berjalan pada perangkat Raspberry Pi 4B.

Dalam penelitian ini perlu dilakukan pelatihan model ML untuk dapat menghasilkan model yang dapat melakukan deteksi produk dengan baik. Untuk melakukan pelatihan tersebut, diperlukan sebuah komputer dengan beberapa *tools*. Pada penelitian kali ini spesifikasi komputer dan *tools* yang digunakan untuk melakukan pelatihan model sesuai dengan Tabel 3.1.

Tabel 3.1 Spesifikasi perangkat yang digunakan untuk pelatihan

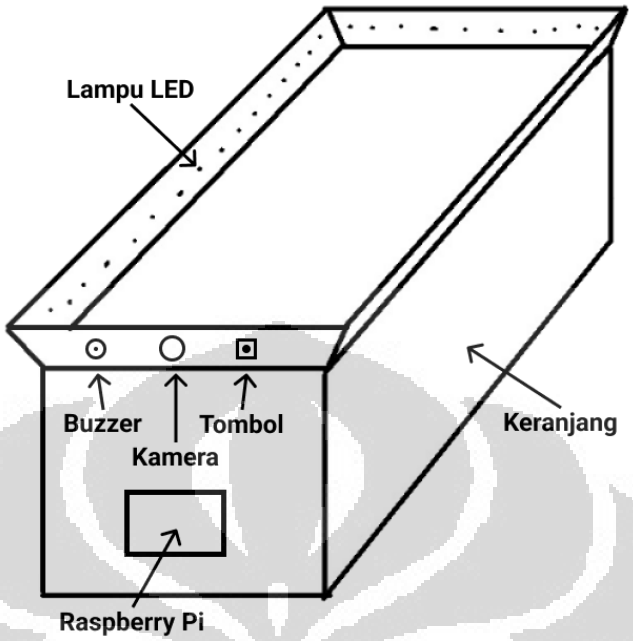
No.	Komponen	Spesifikasi
1	CPU	Intel Xeon CPU 2.20 GHz
2	GPU	NVIDIA Tesla P100-PCIE-16GB
3	RAM	25 GB
4	Python	3.8.7
5	PyTorch	1.11.0 + CUDA v11.5
6	Model	YOLOv5 by Ultralytics versi 6.1

3.1.1. Arsitektur Sistem *Smart Trolley*

Gambar 3.1 Arsitektur Sistem *Smart Trolley*

Pada Gambar 3.1 terdapat ilustrasi arsitektur sistem *smart trolley* secara umum yang akan dibuat pada penelitian ini. Sistem akan dibuat menggunakan sebuah keranjang dengan Raspberry Pi 4B sebagai pusat dari seluruh proses yang akan dilakukan. Tugas utama dari sistem ini adalah melakukan deteksi produk dan mengetahui apakah produk tersebut sedang keluar atau masuk keranjang belanja. Keranjang belanja pada sistem ini hanya bertugas sebagai tempat meletakkan barang seperti umumnya keranjang belanja pada pusat perbelanjaan.

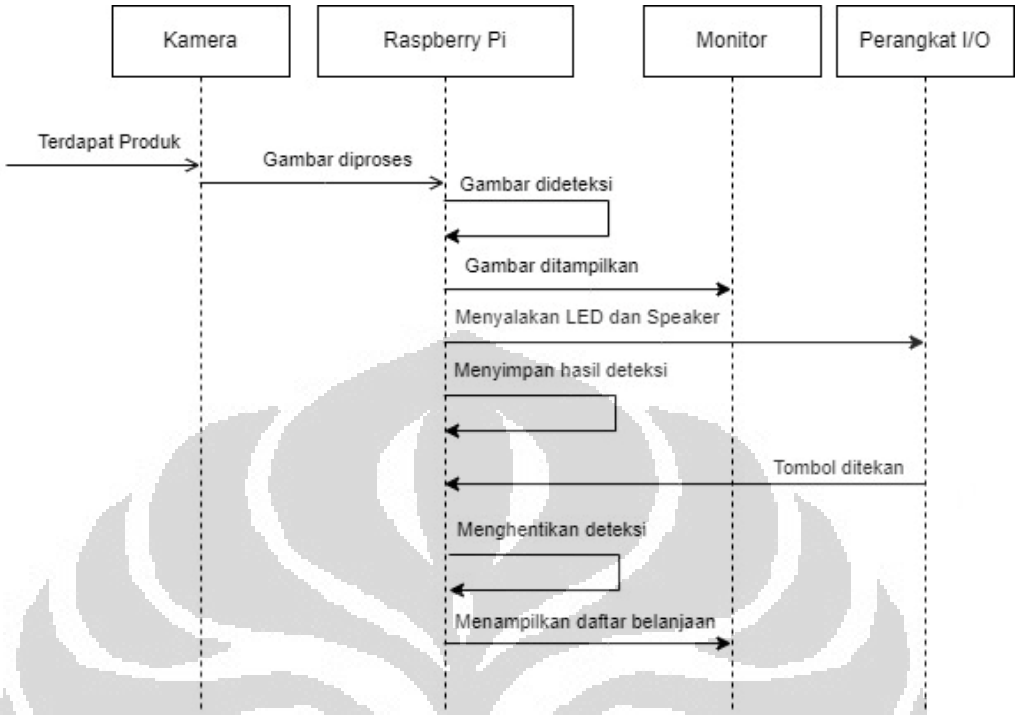
3.1.2. Desain Prototipe



Gambar 3.2 Desain Prototipe Keranjang *Smart Trolley*

Pada Gambar 3.2 terdapat desain dari prototipe yang akan dibuat pada penelitian ini. Pada desain tersebut terdapat keranjang yang telah terintegrasi dengan Raspberry Pi 4B yang terhubung dengan kamera, *buzzer*, tombol, dan lampu. Kamera dibutuhkan untuk dapat menerima masukan berupa gambar yang ada di bagian atas keranjang. *Buzzer* dibutuhkan untuk dapat menandakan jika terdapat produk yang terdeteksi dan untuk membedakan produk yang sedang masuk dengan yang keluar dari keranjang. Tombol dibutuhkan untuk menjalankan dan menghentikan deteksi sekaligus untuk simulasi *checkout* yang akan menampilkan seluruh produk yang ada di dalam keranjang. Lampu LED dibutuhkan sebagai sumber penerangan tambahan untuk membantu kamera menangkap gambar dengan pencahayaan yang lebih baik, lampu tersebut diletakkan di bagian atas mengelilingi keranjang.

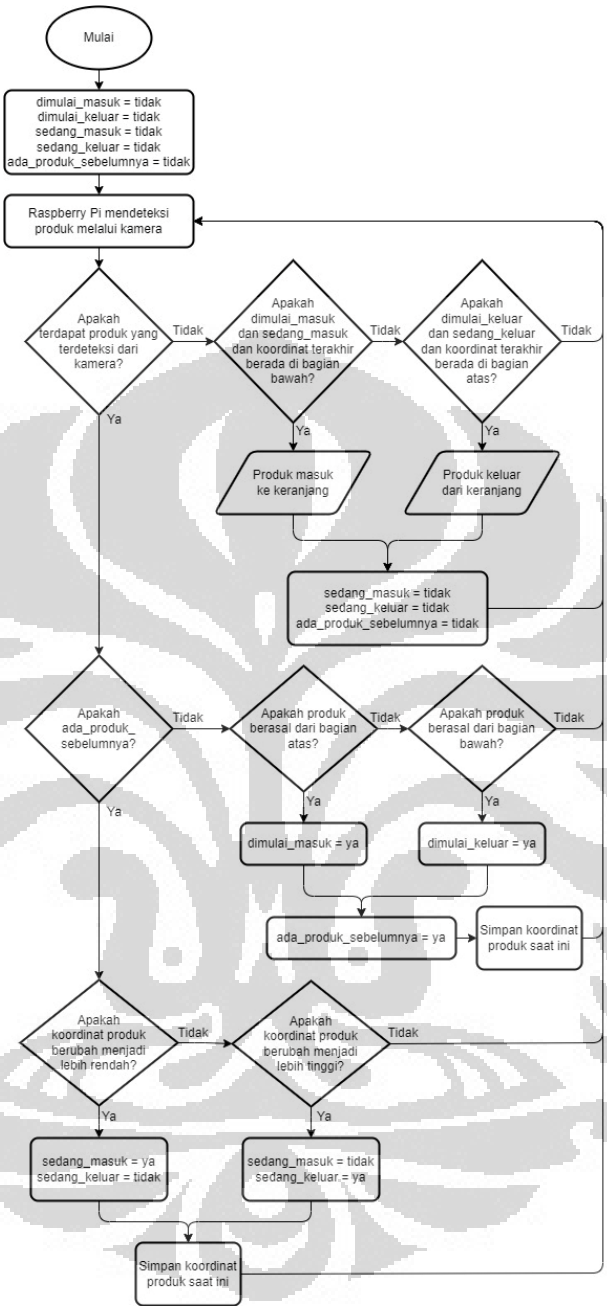
3.1.3. Sekuensial Diagram Sistem *Smart Trolley*



Gambar 3.3 Sekuensial Diagram Sistem *Smart Trolley*

Pada Gambar 3.3 terdapat sekuensial diagram dari keseluruhan sistem yang akan diimplementasikan di dalam Raspberry Pi 4B. Sistem akan menunggu produk hingga berada di depan kamera lalu gambar akan diproses oleh Raspberry Pi 4B untuk mendapatkan gambar yang sesuai dengan yang dibutuhkan oleh model. Setelah gambar siap maka gambar akan diprediksi oleh model yang telah dilatih sebelumnya lalu akan ditampilkan ke monitor. Jika terdapat deteksi maka perangkat akan menghidupkan LED dan *speaker* untuk memberi penanda terdapat produk yang terdeteksi. Setelah terdeteksi maka sistem akan menyimpan hasil deteksi ke dalam variabel untuk nantinya ditampilkan pada saat *checkout*. Saat tombol ditekan maka sistem akan berhenti mendeteksi dan menampilkan seluruh daftar belanjaan ke monitor.

3.1.4. Diagram Alir Sistem *Smart Trolley*

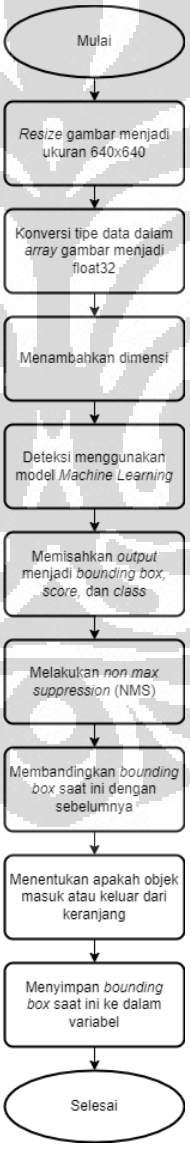


Gambar 3.4 Diagram Alir Sistem *Smart Trolley*

Pada Gambar 3.4 terdapat diagram alir dari keseluruhan sistem yang akan diimplementasikan di dalam Raspberry Pi 4B. Sistem akan melakukan deteksi produk secara *real-time* melalui kamera. Sistem ini saat pertama dijalankan akan menyiapkan beberapa variabel untuk menyimpan berbagai macam kondisi produk saat sedang dideteksi. Setelah itu, sistem akan mendeteksi produk menggunakan

model ML yang sudah dilatih dari *frame* yang ditangkap oleh kamera. Setelah terdapat produk yang terdeteksi, maka akan ditentukan apakah produk tersebut masuk atau keluar dari keranjang belanja dengan memanfaatkan perubahan koordinat lokasi objek yang terdeteksi saat ini dengan yang sebelumnya. Setelah proses untuk mengetahui produk masuk atau keluar keranjang dilakukan, sistem akan kembali mengulang untuk melakukan deteksi objek pada *frame* selanjutnya. Setelah proses berbelanja selesai, seluruh produk yang terdapat di dalam keranjang dapat ditampilkan dengan menekan tombol *finish*.

3.1.5. Diagram Alir *Object Detection*



Gambar 3.5 Diagram Alir Sistem *Object Detection*

Pada Gambar 3.5 terdapat diagram alir sistem deteksi yang akan digunakan dalam penelitian kali ini. Sistem deteksi objek dimulai dengan mengubah ukuran gambar menjadi 640x640 piksel. Lalu tipe data dalam *array* gambar akan diubah menjadi jenis *float* dan perlu menambahkan satu dimensi baru untuk memenuhi keperluan pada *layer convolution*. Setelah gambar diproses, gambar tersebut sudah siap untuk dideteksi menggunakan model ML. Setelah didapat hasil deteksi maka akan dipisahkan menjadi titik koordinat *bounding box* objek, skor, dan kelas objek tersebut. Setelah didapatkan hasil, perlu dilakukan *Non Max Suppression* (NMS) dengan menghitung nilai *Intersection over Union* (IoU) antar *bounding box*. NMS dilakukan untuk memfilter *bounding box* yang terdeteksi sehingga hanya tersisa satu *bounding box* terbaik saja untuk setiap objeknya. Setelah NMS dilakukan, setiap *bounding box* yang terdeteksi akan dibandingkan dengan *bounding box* sebelumnya untuk mengetahui kondisi setiap objek sedang masuk atau keluar dari keranjang. Setelah itu *bounding box* akan disimpan ke dalam variabel untuk menjadi pembanding pada deteksi selanjutnya.

3.2. Langkah Pembuatan Model ML

Gambar 3.6 Alur Pembuatan Model ML

Gambar 3.6 menjelaskan alur yang harus dilakukan pada keseluruhan proses pembuatan dan implementasi model pada penelitian ini. Pembuatan model dimulai dengan melakukan pengumpulan dan pembuatan *dataset* yang berasal dari pengambilan gambar sendiri secara manual. Setelah gambar terkumpul maka data tersebut akan diberikan label sesuai dengan objek apa yang terdapat dalam gambar tersebut. Setelah pelabelan data selesai, dilakukan *augmentasi* pada beberapa gambar untuk memberikan berbagai varian data baru yang akan digunakan untuk pengujian tertentu yang lebih spesifik. Lalu dilakukan pelatihan dan evaluasi model YOLOv5 menggunakan *dataset* yang telah diproses sebelumnya. Saat model sudah siap, maka model perlu dikonversi menjadi TensorFlow Lite supaya bisa diterapkan pada Raspberry Pi 4B. Untuk dapat menjalankan model, diperlukan *Inference* untuk

dapat memasukkan data ke dalam model dan mendapatkan hasil dari model tersebut. Setelah itu, seluruh program sudah bisa digabung menjadi satu di dalam Raspberry Pi 4B dan dijalankan.

3.3. Pengumpulan dan Pembuatan *Dataset*

Proses pengumpulan *dataset* dilakukan dengan menggunakan tiga metode. Metode pertama dilakukan dengan mengambil foto secara manual menggunakan kamera. Metode kedua dilakukan dengan mengambil gambar dari setiap *frame* video produk yang sedang masuk dan keluar keranjang. Metode ketiga dilakukan menggunakan Raspberry Pi 4B dengan berbagai konsisi pencahayaan. *Dataset* akan diambil dari berbagai sisi produk untuk membuat variasi sudut pandang produk.

Dataset pada metode pertama dan kedua akan dibagi menjadi dua bagian, yaitu *training* dan *validation*. Data *training* dan *validation* akan berisi variasi seluruh gambar produk dari *dataset* tergantung pada percobaan yang sedang dilakukan. Pada metode ketiga, *dataset* yang diambil akan dijadikan sebagai data *testing* yang digunakan sebagai *evaluasi* model karena gambar yang diambil akan memberikan gambaran keadaan nyata pada saat model telah diimplementasikan di keranjang belanja pasar swalayan.

3.3.1. Pengumpulan *Dataset* dari Foto Produk Secara Manual

Pengumpulan *dataset* dengan cara mengambil foto produk secara manual dilakukan untuk mendapatkan *dataset* berdasarkan produk yang saat ini dapat ditemukan di toko swalayan. Gambar yang diambil berisi berbagai varian gambar dari masing-masing produk dengan berbagai sisi hingga 360 derajat. Pengambilan gambar secara manual diharapkan dapat memberikan gambaran produk secara detail dengan teknik pengambilan gambar dari berbagai sisi dan sudut produk. *Dataset* yang digunakan pada variasi ini berasal dari penelitian sebelumnya yang membahas hal serupa [28].

3.3.2. Pengumpulan *Dataset* dari *Frame* Video

Pengumpulan *dataset* dengan cara mengambil setiap *frame* dari video produk yang bergerak dilakukan untuk mendapatkan *dataset* dengan *motion blur* yang berasal dari keadaan riil. *Motion blur* diperlukan untuk memberikan efek yang seharusnya ada pada saat produk sedang masuk atau keluar dari keranjang. Untuk setiap produknya, video yang diambil harus menampilkan berbagai sisi produk yang sedang masuk dan keluar keranjang dengan minimal terdapat enam sisi produk. Pengambilan *dataset* dari *frame* video diharapkan dapat meningkatkan akurasi model terutama untuk dapat mendeteksi produk yang sedang bergerak. Pada Gambar 3.7 terdapat ilustrasi foto yang akan diambil menggunakan teknik ini.

Gambar 3.7 Ilustrasi *dataset* dari *frame* video

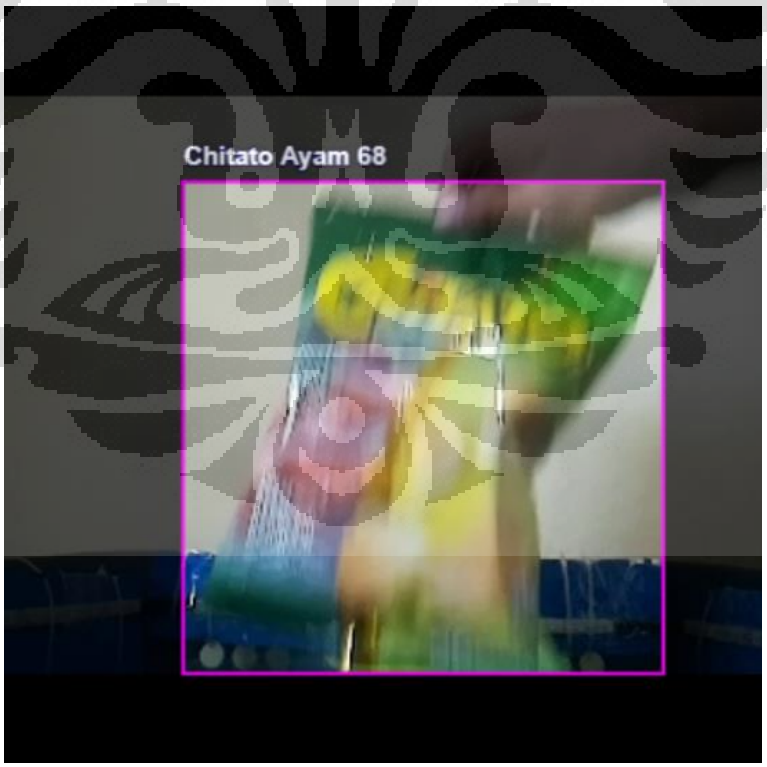
3.3.3. Pengumpulan *Dataset* Pengujian

Pengumpulan *dataset* untuk proses pengujian dilakukan dengan cara yang sama seperti Subbab 3.3.2, namun *dataset* akan diambil menggunakan kamera yang terhubung dengan Raspberry Pi 4B. Hal tersebut dilakukan untuk menguji model

dengan menggunakan gambaran keadaan sistem yang telah diimplementasikan pada Raspberry Pi 4B. *Dataset* pengujian akan diambil dengan berbagai kondisi pencahayaan untuk mendapatkan beberapa skenario keadaan yang mungkin terjadi pada saat sistem telah diimplementasikan pada keadaan riil.

3.4. Pengolahan *Dataset*

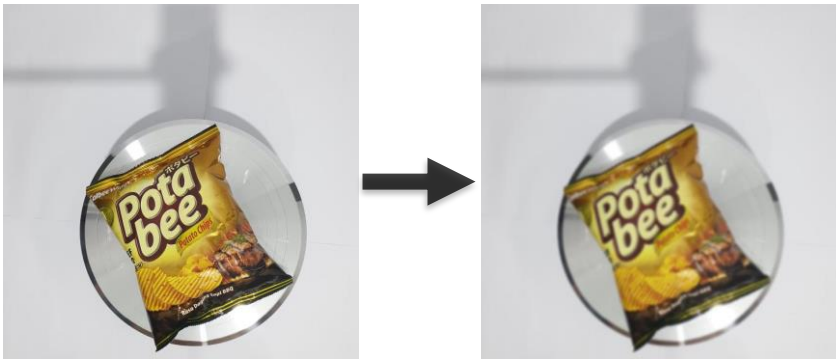
Proses pengolahan *dataset* dilakukan untuk mempersiapkan *dataset* supaya dapat digunakan dengan baik pada proses *training* dan evaluasi menggunakan model YOLOv5. Ketentuan yang harus terdapat pada *dataset* sebelum digunakan adalah anotasi gambar yang menentukan titik di mana produk berada dalam gambar. Untuk setiap gambar terdapat satu *file* anotasi yang berisi data seluruh *bounding box* pada gambar tersebut dengan setiap *bounding box* yang dipisahkan menggunakan baris baru. Proses anotasi tersebut akan dilakukan *tools* yang bernama Roboflow untuk menghasilkan format yang sesuai dengan yang dibutuhkan oleh YOLOv5. Contoh gambar yang sudah dianotasi dapat dilihat pada Gambar 3.8.



Gambar 3.8 Gambar *Dataset* yang sudah dianotasi

Selain melakukan anotasi gambar, gambar juga perlu diubah resolusinya untuk menyesuaikan *input* yang diperlukan model, dengan resolusi 640x640 untuk YOLOv5 dengan *backbone original* dan 224x224 untuk YOLOv5 dengan *backbone swin-transformer*. Karena aspek rasio gambar yang dibutuhkan adalah 1:1 sedangkan beberapa gambar yang diambil dari kamera memiliki aspek rasio yang berbeda, maka gambar perlu dipotong untuk menyesuaikan aspek rasionya. Pemotongan gambar dilakukan menggunakan Python dengan bantuan *library* Pillow. Roboflow tidak digunakan pada proses pemotongan ini karena diperlukan penyesuaian bagian yang dipotong pada beberapa gambar yang memiliki objek di bagian pinggirnya, sedangkan fitur yang terdapat pada Roboflow hanya dapat memotong gambar pada bagian tengah saja. Setelah gambar dipotong, maka akan digunakan Roboflow untuk melakukan penyesuaian resolusi gambar sesuai *input* yang diinginkan model. Setelah itu juga harus dilakukan pembagian *dataset* di Roboflow menjadi *train* dan *valid* yang akan dilakukan secara manual karena dibutuhkan pembagian gambar berdasarkan objek dan sudut pandang yang sesuai pada masing-masing bagian.

Untuk menciptakan variasi *dataset* pada eksperimen, perlu dilakukan berbagai *augmentasi* sesuai dengan kebutuhan. Proses *augmentasi* yang akan dilakukan adalah memberikan Gaussian Blur pada *dataset* statis dengan menggunakan bantuan *library* Pillow. *Augmentasi* tidak dilakukan langsung pada Roboflow karena adanya limitasi pada Roboflow yang hanya dapat melakukan *augmentasi* pada data *training* saja, sedangkan pada beberapa percobaan dibutuhkan *augmentasi* pada data testing. Hasil *augmentasi blur* yang dilakukan pada *dataset* statis menghasilkan jumlah dua kali dari gambar awal sehingga total jumlah gambar yang akan digunakan pada *dataset* statis *blur* adalah sejumlah 3990 *dataset training* dan 1050 *dataset* validasi. Contoh gambar sebelum dan sesudah diaugmentasi adalah seperti pada Gambar 3.9.



Gambar 3.9 Gambar *Dataset* yang diaugmentasi

Dataset yang sudah diproses dapat diekspor ke dalam berbagai format, salah satunya adalah format YOLOv5. Pada Roboflow juga terdapat fitur untuk langsung mengunduh berkas *zip dataset* ke komputer atau mendapatkan *link zip dataset* dengan memanfaatkan Roboflow sebagai tempat untuk menyimpan *dataset* yang sudah diolah. Fitur untuk menyimpan *dataset* pada Roboflow membuat *dataset* dapat diakses langsung secara *online* kapan saja dan dari mana saja. Fitur tersebut dapat mempermudah penggunaan *dataset* karena Google Colab bisa langsung mengunduh *dataset* dan tidak perlu lagi melakukan *upload dataset* ke Google Colab terlebih dahulu yang memakan waktu cukup lama. Penelitian kali ini akan memanfaatkan fitur penyimpanan *dataset* secara *online* pada Roboflow untuk mempermudah penggunaan *dataset*. Google Colab akan melakukan pengunduhan berkas *zip* dan melakukan *unzip* berkas yang telah diunduh dengan menggunakan kode seperti pada Gambar 3.10.

```
dataset_url = "https://app.roboflow.com/ds/QV1aexu7Ai?key=zNX5qB0JdR"
!curl -L $dataset_url > dataset.zip
!unzip dataset.zip -d dataset
```

Gambar 3.10 Kode untuk melakukan *download dataset* dari Roboflow

3.5. Pelatihan Model

Proses pelatihan atau *training* model dilakukan menggunakan *dataset* yang sudah disiapkan dan dibagi menjadi *training*, *validation*, dan *testing*. Pada penelitian ini, model yang digunakan untuk melakukan deteksi objek adalah YOLOv5 yang dikembangkan oleh Ultralytics dan juga variasi dari YOLOv5 yang menggunakan

backbone swin-transformer. Model YOLOv5 adalah model *object detection* yang bersifat *open source* dan dapat ditemukan di *repository* Github milik Ultralytics. YOLOv5 dipilih karena memiliki tingkat efisiensi yang sangat tinggi dibandingkan dengan model *object detection* lainnya berdasarkan hasil *training* pada *dataset* COCO. YOLOv5 dapat dijalankan dengan performa yang lebih cepat namun memiliki tingkat akurasi yang lebih tinggi. Jenis YOLOv5 yang digunakan pada penelitian ini adalah YOLOv5n karena dapat dijalankan dengan 4.5 gigaflops dan masih memenuhi spesifikasi Raspberry Pi 4B [16]. Penggunaan variasi *backbone* bertujuan untuk mengetahui performa dan akurasi YOLOv5 jika menggunakan *backbone* yang berbeda dari aslinya.

```
python train.py --cfg models/yolov5n.yaml --weights yolov5n.pt --data trolle/data.yaml
--epochs 500 --device 0 --batch-size 2
```

Gambar 3.11 Perintah untuk melakukan pelatihan model

Proses *training* model dapat dilakukan dengan menggunakan *file* yang sudah disediakan dengan nama train.py. Training dapat dijalankan menggunakan perintah seperti yang terdapat pada Gambar 3.11 dengan parameter cfg untuk menentukan lokasi konfigurasi model, weights untuk menentukan *pre-trained weights* yang ingin digunakan jika tersedia, data untuk menentukan lokasi konfigurasi dataset yang digunakan, epoch untuk menentukan jumlah *epoch*, device untuk menentukan perangkat GPU yang digunakan jika tersedia, batch-size untuk menentukan ukuran *batch* dari *dataset* yang mana semakin besar *batch* yang digunakan maka akan semakin banyak memori yang dikonsumsi.

3.6. Evaluasi Model

Proses *training* dilakukan bersamaan dengan evaluasi model pada setiap *epoch*-nya. Setelah proses *training* selesai maka akan terdapat file yang berisi berbagai diagram yang menampilkan nilai *F1-Score, Precision, Recall,* serta *mAP* yang dapat digunakan sebagai bahan evaluasi dari *training* yang sudah dilakukan. Selain itu terdapat juga *confusion matrix* yang menampilkan hasil deteksi model untuk setiap objeknya dari validasi yang dilakukan. *Confusion matrix* yang dihasilkan

menggunakan parameter deteksi *default* dengan Confidence (batas minimum akurasi) 0.25 dan IoU Threshold 0.45.

3.7. Konversi Model menjadi TensorFlow Lite

Konversi model diperlukan untuk mempermudah proses *deployment* yang akan dilakukan ke perangkat Raspberry Pi 4B. Selain itu, model yang berjenis Tensorflow Lite juga dapat memperkecil ukuran model untuk dapat berjalan dengan lebih cepat tanpa mengurangi akurasi dengan signifikan [29]. Konversi model dilakukan setelah model selesai menjalankan *training* dan sudah siap untuk digunakan pada tahap *deployment*.

```
python export.py --data trolle/data.yaml --weights runs\train\exp\weights\best.pt
--include tflite
```

Gambar 3.12 Perintah untuk melakukan konversi model

Proses konversi model dapat dilakukan dengan menggunakan *file* yang sudah disediakan dalam YOLOv5 dengan nama export.py. Konversi model dapat dijalankan dengan menggunakan perintah seperti yang terdapat pada Gambar 3.12 dengan parameter data untuk menentukan lokasi konfigurasi dataset yang digunakan, weights untuk menentukan lokasi *weights* dari hasil *training* yang sudah dilakukan sebelumnya, dan include untuk menentukan konversi model menjadi jenis apa yang mana pada penelitian ini *output* model yang diinginkan adalah TensorFlow Lite.

3.8. Pembuatan TensorFlow Lite *Inference*

Model yang telah dikonversi menjadi jenis TensorFlow Lite perlu dilakukan *inference* untuk dapat menggunakannya. *Inference* berfungsi untuk mengolah gambar lalu memasukkannya pada model untuk mendapatkan keluarannya lalu mengolahnya supaya dapat digunakan dengan mudah.

```
def detector(frame, model_path, input_resolution, output_resolution, labels, confidence,
iou_threshold):
    # Input processing
    img = input_processing(frame, input_resolution)

    # initialize model
    interpreter = tf.lite.Interpreter(model_path=model_path)
    interpreter.allocate_tensors()

    # Get input and output tensors
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()

    # set the image into the model
    interpreter.set_tensor(input_details[0]['index'], img)

    # run model
    interpreter.invoke()

    # get results
    output = interpreter.get_tensor(output_details[0]['index'])[0]

    # Output processing
    output = output_processing(output, output_resolution/input_resolution)

    # NMS
    max_output_size = 100
    obj_nms = tf.image.non_max_suppression(output[1], output[2], max_output_size,
iou_threshold=iou_threshold)

    # show the output
    return show_output(frame, output_resolution, output, obj_nms, labels, confidence)
```

Gambar 3.13 Fungsi untuk melakukan deteksi

Pada Gambar 3.13 terdapat fungsi yang dapat digunakan untuk melakukan deteksi produk dari gambar yang diberikan. Fungsi ini memerlukan *input* parameter berupa *array* gambar, lokasi model, ukuran resolusi *input* model, dan ukuran resolusi gambar dari hasil. Fungsi ini melakukan pemanggilan *interface* yang sudah disediakan oleh TensorFlow Lite untuk dapat mengambil konfigurasi model. Setelah memasukkan model maka fungsi akan memasukkan *input* gambar ke dalam model dan menyimpan hasil deteksi. Setelah mendapatkan hasil deteksi maka fungsi ini akan melakukan NMS untuk mendapatkan *bounding box* akhir yang sudah tidak terduplikasi.

```
def input_processing(frame, input_resolution):
    # Resize image
    img = cv2.resize(frame, input_resolution, interpolation = cv2.INTER_AREA)
    # Change data type to float
    img = img.astype('float32')
    # Change convert to 0-1
    img /= 255
    # Expand dimensions
    img = np.expand_dims(img, axis=0)
    return img
```

Gambar 3.14 Fungsi untuk memproses gambar *input*

Pada Gambar 3.14 terdapat fungsi yang digunakan untuk memproses *input array* gambar. Fungsi ini memerlukan *input* parameter berupa *array* gambar dan resolusi *input* model. Fungsi ini akan melakukan *resize* gambar sesuai dengan ukuran yang diperlukan oleh input model. Setelah itu tipe data dalam *array* gambar diubah menjadi tipe *float* dan diskalasi menjadi nilai 0 sampai 1. Lalu tahap terakhir pemrosesan gambar adalah dengan menambahkan satu dimensi tambahan.

```
def output_processing(output, output_resolution):
    # Scaling the boxes
    imgsz = output_resolution
    output[..., 0] *= imgsz[1]  # x
    output[..., 1] *= imgsz[0]  # y
    output[..., 2] *= imgsz[1]  # w
    output[..., 3] *= imgsz[0]  # h
    center = output[..., :4]

    # Convert nx4 boxes from [x, y, w, h] to [x1, y1, x2, y2]
    # where xy1=top-left, xy2=bottom-right
    bbox = np.zeros((len(output), 4))
    bbox[..., 0] = output[:, 0] - output[:, 2] / 2
    bbox[..., 1] = output[:, 1] - output[:, 3] / 2
    bbox[..., 2] = output[:, 0] + output[:, 2] / 2
    bbox[..., 3] = output[:, 1] + output[:, 3] / 2

    # Get scores and classes
    scores = output[..., 4]
    classes = output[..., 5:]

    return center, bbox, scores, classes
```

Gambar 3.15 Fungsi untuk memproses hasil *output* model

Pada Gambar 3.15 terdapat fungsi yang digunakan untuk memproses hasil dari deteksi objek yang dilakukan oleh model. Fungsi ini memerlukan *input* parameter

berupa hasil deteksi dan resolusi *output* gambar yang diinginkan. Fungsi ini akan melakukan *scaling* hasil *bounding box* sesuai dengan resolusi *output* yang diinginkan. Lalu fungsi ini akan mengubah format *bounding box* dari YOLOv5 yang berbentuk koordinat x, koordinat y, lebar, dan panjang menjadi koordinat x1, y1, x2, y2. Lalu fungsi ini akan memecah setiap *output* deteksi YOLOv5 dari hanya satu variabel menjadi variabel *bounding box*, *score*, dan *class*.

```
def show_output(frame, output_resolution, output, obj_nms, lbl, confidence):
    center, bbox, scores, classes = output
    retval = []

    # Resize images
    img = cv2.resize(frame, output_resolution, interpolation=cv2.INTER_AREA)

    best = 0
    best_index = -1
    for i in obj_nms:
        # Filter results
        if scores[i] > confidence and scores[i] > best:
            best = scores[i]
            best_index = i

    xmin = int(max(1, bbox[best_index, 0]))
    ymin = int(max(1, bbox[best_index, 1]))
    xmax = int(min(output_resolution[1], bbox[best_index, 2]))
    ymax = int(min(output_resolution[0], bbox[best_index, 3]))

    # Get class
    clss = np.argmax(classes[best_index])

    retval.append([clss, center[best_index], [xmin, ymin, xmax, ymax]])

    # Create bounding box
    cv2.rectangle(img, (xmin, ymin), (xmax, ymax), (0, 255, 0), 4)
    cv2.putText(img, lbl[clss] + " %.2f" % scores[best_index], (xmin, ymin - 10),
0, 1, (255, 0, 0), 2)

    return img, retval
```

Gambar 3.16 Fungsi untuk menampilkan hasil deteksi

Pada Gambar 3.16 terdapat fungsi yang digunakan untuk menampilkan gambar hasil akhir dari deteksi objek yang telah dilakukan. Fungsi ini memerlukan *input* parameter berupa *array* gambar, resolusi gambar *output*, hasil *output*, dan objek deteksi hasil NMS. Fungsi ini akan melakukan *resize* gambar sesuai dengan resolusi *output* yang diinginkan. Lalu seluruh objek yang telah melalui NMS akan difilter dengan skor diatas 50% dan dibawah 100%. Lalu seluruh objek yang memenuhi

kriteria akan digambar *bounding box*-nya beserta label dan akurasinya di atas gambar yang dideteksi.

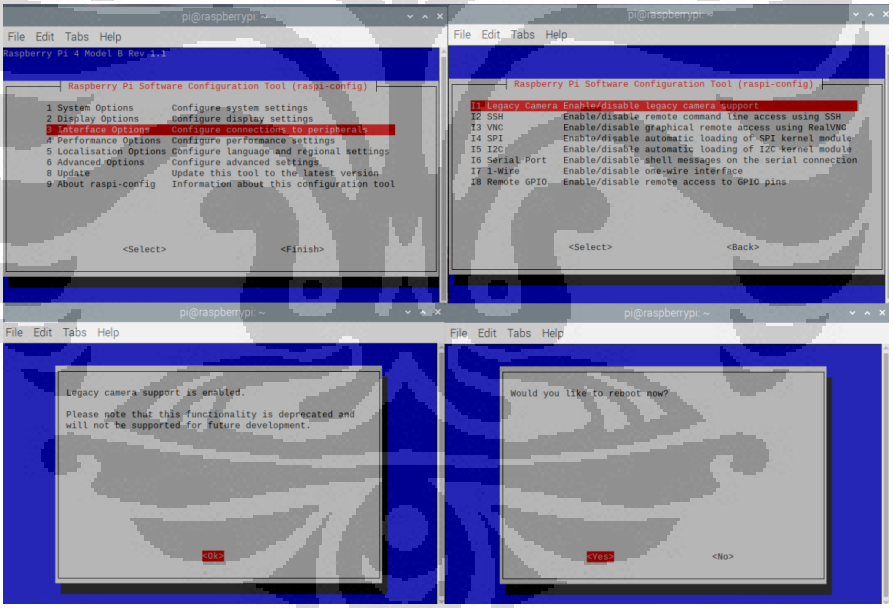
```
detect = self.detected
if detect and curr_time - detect[4] > self.obj_lost_intrvl:
    begin_cond = bool(detect[3] & 0b10)
    end_cond = bool(detect[3] & 0b01)
    if begin_cond and end_cond and detect[5][3] > self.out_res[0]/2:
        self.product_in_trolley.remove(self.lbl[detect[0]])
    elif not begin_cond and not end_cond and detect[5][1] < self.out_res[0]/2:
        self.product_in_trolley.append(self.lbl[detect[0]])
    self.detected = []
result = detector(
            frame,
            self.model_path,
            self.in_res,
            self.out_res,
            self.lbl,
            self.conf,
            self.iou_th
        )
img = result[0]
for data in result[1]:
    classes = data[0]   # Class hasil deteksi
    center = data[1]    # Titik tengah dari objek yang terdeteksi (x, y, w, h)
    bbox = data[2]      # Bounding box hasil deteksi
    x1 = bbox[0]        # Koordinat x kiri atas objek hasil deteksi
    y1 = bbox[1]        # Koordinat y kiri atas objek hasil deteksi
    x2 = bbox[2]        # Koordinat x kanan bawah objek hasil deteksi
    y2 = bbox[3]        # Koordinat y kanan bawah objek hasil deteksi

    detect = self.detected

    if not detect:
        if y1 < self.out_res[0]/2:
            self.tmp_status = "Mulai Masuk"
            self.detected = [classes, center[0], center[1], 0b00, curr_time, bbox]
        elif y2 > self.out_res[0]/2:
            self.tmp_status = "Mulai Keluar"
            self.detected = [classes, center[0], center[1], 0b10, curr_time, bbox]

    elif classes == detect[0]:
        detect[4] = curr_time
        detect[5] = bbox
        delta_bbox = [center[0]-detect[1], center[1]-detect[2]] # [x, y]

        detect[1:3] = center[:2]
        if delta_bbox[1] > 0:
            self.tmp_status = "Lagi Masuk"
            detect[3] &= 0b10
        else:
            self.tmp_status = "Lagi Keluar"
            detect[3] |= 0b01
```

Gambar 3.17 Fungsi untuk melakukan penangkapan *frame* dari *Webcam* dan deteksi

Pada Gambar 3.17 terdapat fungsi yang digunakan untuk melakukan uji coba pada perangkat yang terdapat *Webcam* atau kamera yang terhubung dengannya. Pada funsi tersebut juga melakukan fungsi untuk mengetahui apakah produk masuk atau keluar keranjang dengan cara menyimpan dan membandingkan data saat produk pertama kali terderteksi dan terakhir kali terdeteksi. Produk akan dinyatakan telah berhasil masuk atau keluar dari keranjang jika telah tidak terdeteksi dan melewati batas waktu yang telah ditentukan.

3.9. Deployment

Proses *deployment* akan menerapkan model yang sudah dilakukan *training* dan konversi model ke dalam Raspberry Pi 4B. Pada proses ini Raspberry Pi 4B harus dipasang beberapa *library* untuk mendukung proses deteksi objek yang akan dilakukan antara lain TensorFlow, NumPy, OpenCV, dan Matplotlib.



Gambar 3.18 Konfigurasi kamera pada Raspberry Pi 4B

Setelah itu perlu dilakukan konfigurasi kamera dengan mengaktifkannya melalui panel konfigurasi Raspberry Pi 4B. Pada Gambar 3.18 terdapat alur konfigurasi melalui panel konfigurasi Raspberry Pi 4B dengan tahapan pertama memasuki Interface Options lalu Legacy Camera dan akan muncul dialog yang menampilkan

informasi bahwa kamera telah diaktifkan lalu Raspberry Pi 4B memerlukan *reboot* untuk menerapkan konfigurasinya.

Pada proses *deployment* ini, *inference* yang sudah dibuat sebelumnya beserta modelnya dipindahkan ke dalam Raspberry Pi 4B. *Inference* dikonfigurasi untuk dapat mengambil gambar menggunakan modul picamera yang ada pada Raspberry Pi 4B. *Inference* dapat langsung dijalankan pada Raspberry Pi 4B menggunakan Python untuk mendeteksi produk dan mendapatkan hasil dari deteksi yang dilakukan.

BAB 4
IMPLEMENTASI DAN PEMBAHASAN

Bab ini membahas implementasi, pengujian, dan analisa terhadap model YOLOv5 yang dijalankan pada Raspberry Pi 4B dalam melakukan deteksi produk FMCG secara *real-time*. Penelitian ini difokuskan untuk merancang model YOLOv5 dan membuat *dataset* yang paling sesuai untuk melakukan deteksi produk FMCG dengan keterbatasan sistem yang harus dapat diterapkan pada *smart trolley*.

Untuk mencapai tujuan penelitian ini, dilakukan tiga pengujian terhadap model, *dataset*, dan intensitas cahaya. Pengujian terhadap model dilakukan dengan mengganti *backbone* supaya didapatkan model yang paling efisien. Pengujian terhadap *dataset* dilakukan dengan membandingkan *dataset* gambar diam dengan *dataset* gambar bergerak untuk mendapatkan jenis *dataset* yang terbaik bagi sistem *smart trolley*. Pengujian intensitas cahaya dilakukan dengan memberikan variasi sumber cahaya untuk mengetahui pengaruh penambahan lampu di sekitar troli.

Terdapat juga dua pengujian sistem yang akan menguji performa keseluruhan sistem yang sudah dibuat untuk mendeteksi produk yang sedang keluar dan masuk ke dalam troli. Pengujian tersebut adalah pengujian dengan menggunakan variasi kecepatan produk yang masuk dan keluar untuk mengetahui tingkat keberhasilan dari sistem dalam mendeteksi produk. Lalu pengujian terakhir adalah untuk mendeteksi produk yang keluar dan masuk dengan menggunakan sampling gambar yang lebih dari satu untuk sekali melakukan deteksi untuk mengetahui potensi sistem untuk mendeteksi produk yang keluar dan masuk dengan lebih *real-time*. Seluruh pengujian yang dilakukan akan menggunakan perangkat komputer pengujian dan perangkat Raspberry Pi 4B untuk mengetahui performa maksimal model dan performa pada perangkat SBC.

4.1. Spesifikasi Perangkat Pengujian

Dalam melakukan pengujian sistem, digunakan perangkat komputer pengujian dan Raspberry Pi 4B yang digunakan untuk melakukan deteksi menggunakan model

yang telah dibuat. Pada pengujian ini spesifikasi perangkat yang akan digunakan adalah seperti pada Tabel 4.1.

Tabel 4.1 Spesifikasi perangkat pengujian yang digunakan

No.	Komponen	Spesifikasi
1	CPU	Intel Core i3-1115G4 CPU 4.10 GHz
2	GPU	NVIDIA GeForce MX450 2GB GDDR6
3	RAM	16 GB DDR4-3200
4	Sistem Operasi	Windows 11
5	Python	3.9.5
6	PyTorch	1.11.0 + CUDA v11.5
7	*Embedded Device*	Raspberry Pi 4B RAM 4 GB
8	Resolusi kamera	5 MP

4.2. *Dataset* Pelatihan dan Pengujian

Dataset yang digunakan pada penelitian ini diambil menggunakan beberapa jenis gambar, yaitu pengumpulan *dataset* dari foto produk secara manual yang akan disebut sebagai *dataset* statis, pengumpulan *dataset* dari *frame* video yang akan disebut sebagai *dataset* bergerak, serta pengumpulan *dataset* pengujian.

Dataset statis merupakan *dataset* yang didapatkan dari penelitian sebelumnya yang melakukan penelitian serupa menggunakan *dataset* dengan gambar yang tidak bergerak [28]. *Dataset* statis akan digunakan dengan *augmentasi* yang akan menambahkan efek *blur* ke dalam gambar atau akan disebut sebagai *dataset* statis *blur*. *Dataset* yang telah ditambahkan *augmentasi blur* akan digunakan sebagai *dataset* pembanding pada percobaan kedua.

Untuk *dataset* bergerak diambil menggunakan kamera *handphone* dengan menggunakan fitur *slow mode* yang membuat produk lebih terlihat jelas untuk setiap *frame*-nya. Intensitas cahaya pada saat pengambilan *dataset* adalah sebesar 200 lux. Video yang diambil adalah video dari produk yang dijatuhkan dengan

kecepatan awal 0 m/s dari ketinggian 70 cm. Video yang telah diambil akan dikonversi menjadi gambar untuk setiap *frame*-nya dengan menghapus frame yang tidak dibutuhkan dan hanya menyisakan *frame* yang terdapat produk di dalamnya. Untuk pengujian kedua akan diambil dari kombinasi *dataset* bergerak dengan *dataset* statis *blur* atau yang akan disebut sebagai kombinasi *dataset* bergerak dengan statis *blur*.

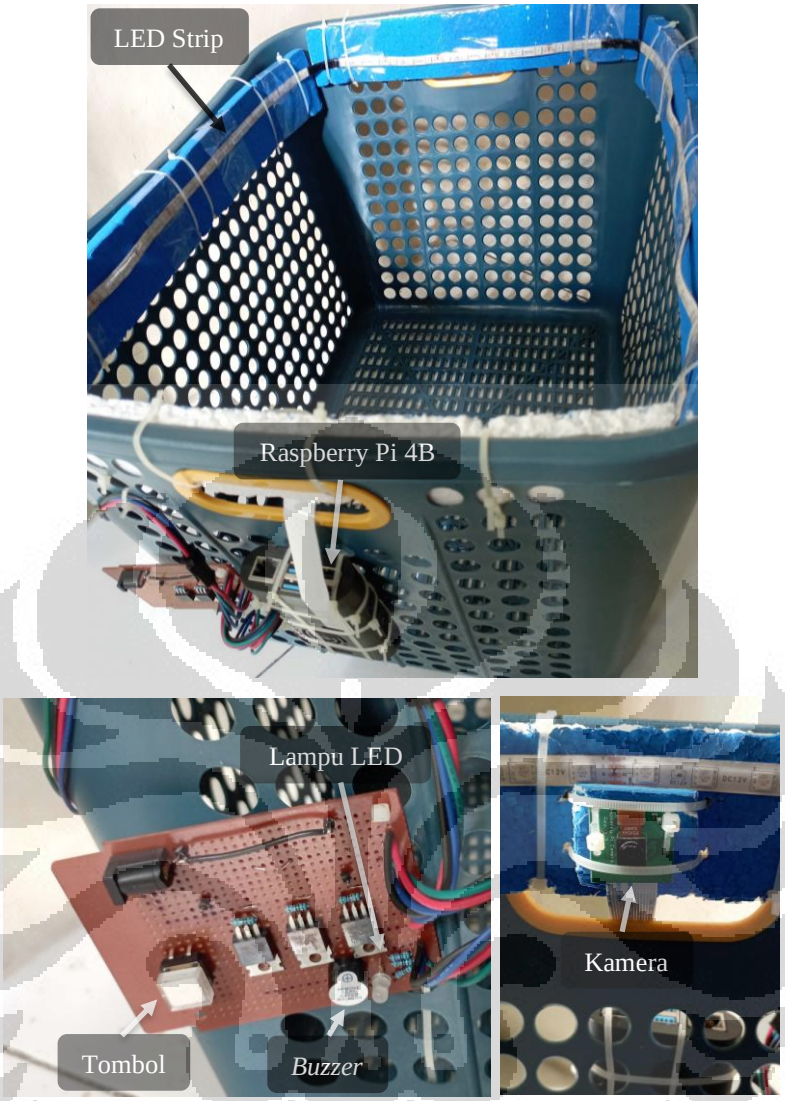
Untuk *dataset* pengujian akan diambil menggunakan kamera yang terhubung dengan Raspberry Pi 4B dengan resolusi 5-megapixel. *Dataset* ini juga diambil dari video produk yang bergerak lalu dikonversi menjadi gambar untuk setiap *frame*-nya. Untuk produk yang diambil akan digerakkan masuk dan keluar dari troli menggunakan tangan dengan kecepatan yang acak. *Dataset* ini dibuat untuk digunakan pada pengujian ketiga yang ingin membuktikan pengaruh intensitas cahaya terhadap akurasi model. Pada *dataset* ini terdapat tiga jenis gambar yang akan diambil dibedakan berdasarkan intensitas cahayanya. Intensitas cahaya yang akan digunakan adalah cahaya yang bersumber dari matahari yang redup dengan intensitas cahaya 100 lux, cahaya yang bersumber dari lampu ruangan dengan intensitas cahaya 40 lux, dan cahaya yang bersumber dari lampu ruangan ditambah lampu di sekitar troli dengan intensitas cahaya 80 lux.

Pada Gambar 4.1 terdapat ilustrasi dari teknik pengambilan *dataset* yang dilakukan untuk mengambil video pada *dataset* bergerak dan *dataset* pengujian. Pengambilan *dataset* dilakukan langsung di depan troli untuk memberikan gambaran yang cukup tepat saat lampu di sekitar troli dihidupkan.

Gambar 4.1 Teknik pengambilan gambar *dataset* bergerak

4.3. Implementasi Sistem

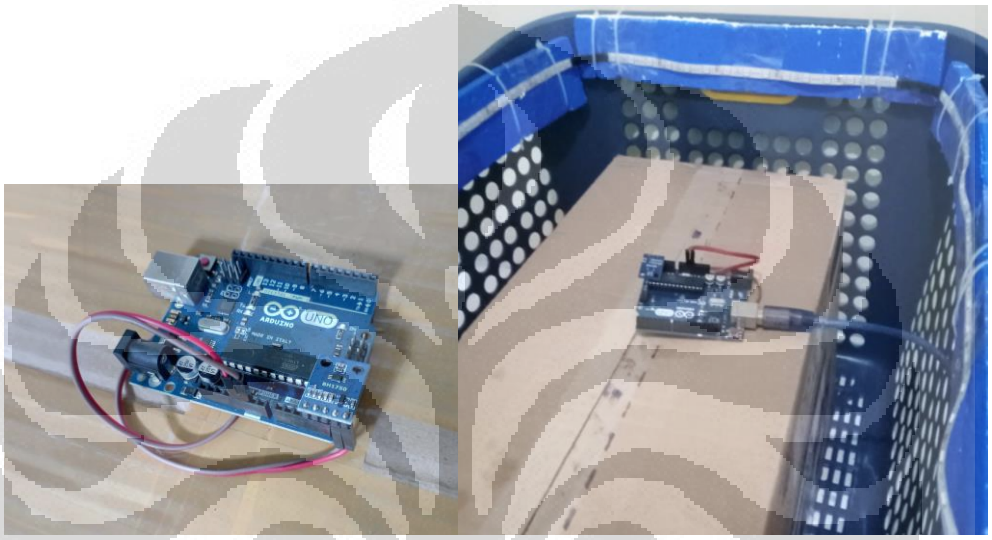
Penelitian ini dirancang untuk dapat berjalan pada Raspberry Pi 4B yang dipasang pada prototipe keranjang. Model yang dibuat dan dilatih menggunakan PyTorch akan dikonversi menjadi jenis TensorFlow Lite supaya dapat berjalan dengan baik pada perangkat Raspberry Pi 4B. Model akan diimplementasikan ke Raspberry Pi 4B menggunakan *inference custom* dengan tambahan algoritma yang berfungsi untuk mengetahui arah pergerakan dari objek yang sedang terdeteksi. Untuk dapat menjalankan model dengan baik, perangkat Raspberry Pi 4B harus sudah terpasang beberapa *library* seperti PyTorch, Matplotlib, OpenCV, Numpy, dan beberapa *library* pendukung lainnya.



Gambar 4.2 Prototipe Sistem *Smart Trolley*

Pada Gambar 4.2 dapat dilihat implementasi sistem *smart trolley* menggunakan prototipe yang dibuat menggunakan keranjang dengan ukuran panjang 56 cm, lebar 44 cm, dan tinggi 33 cm. Pada keranjang tersebut terdapat Raspberry Pi 4B yang digunakan sebagai pusat pemrosesan sistem dan telah terintegrasi dengan kamera, LED Strip, dan PCB kontrol sistem. Lampu LED dan *buzzer* yang berada pada PCB kontrol sistem berfungsi untuk memberikan tanda pada saat proses deteksi. Lampu LED akan menyala dengan warna putih jika terdapat produk yang sedang terdeteksi, menyala dengan warna hijau jika terdapat produk yang berhasil masuk ke dalam keranjang, dan menyala dengan warna merah jika terdapat produk yang berhasil keluar dari keranjang. *Buzzer* akan berbunyi sekali jika terdapat produk yang

berhasil masuk atau keluar dari dalam keranjang. LED Strip yang terdapat di sekeliling keranjang berfungsi untuk membantu memberi penerangan tambahan pada saat proses deteksi. LED Strip akan menyala pada saat sistem sedang berjalan dan mati saat sistem dihentikan. Sistem dapat dijalankan ataupun dihentikan dengan cara menekan tombol yang terdapat pada PCB kontrol sistem. Saat sistem dihentikan maka daftar barang yang ada di dalam keranjang akan ditampilkan melalui terminal yang menjalankan sistem deteksi pada Raspberry Pi 4B.



Gambar 4.3 Alat pengukur intensitas cahaya dan lokasi penempatannya

Pada Gambar 4.3 terdapat alat yang digunakan untuk melakukan pengukuran tingkat intensitas cahaya guna membantu pengujian yang akan dilakukan. Alat tersebut menggunakan sensor BH1750 yang dapat membaca intensitas cahaya antara 1 - 65535 lx dan memiliki akurasi ±80% [18]. Hasil pengukuran tersebut akan dibaca oleh Arduino dan ditampilkan melalui komunikasi serial ke komputer pengujian. Sensor ini akan diletakkan di tengah keranjang dengan jarak antara ujung sensor dengan ujung keranjang sejauh 20 cm dan 27 cm.

4.4. Skenario Pengujian

Terdapat tiga jenis pengujian yang dilakukan pada penelitian kali ini, yaitu pengujian terhadap variasi *backbone* model, jenis *dataset* yang berbeda, dan kondisi pencahayaan yang berbeda. Seluruh pengujian akan dilakukan pada dua perangkat yaitu komputer pengujian dan Raspberry Pi 4B dengan spesifikasi seperti yang telah

dijelaskan pada Subbab 4.1. Pengujian dilakukan menggunakan *inference* yang sama untuk setiap perangkat dan setiap model. Setiap pengujian akan dilakukan sebanyak lima kali untuk mendapatkan data hasil yang lebih valid dari satu kali pengujian.

Pengujian pertama dilakukan dengan variasi *backbone* model untuk mengetahui pengaruh penggunaan *backbone* yang berbeda pada model YOLOv5 dan mencari *backbone* yang memiliki akurasi tinggi dan dapat mendeteksi produk secara *realtime* dengan baik. *Backbone* yang digunakan adalah *backbone* asli dari model yolov5n dan *backbone* dari Swin Transformer. Pada pengujian ini akan digunakan nilai *F1 Score* dan mAP dari model untuk menjadi acuan seberapa bagus model dalam melakukan deteksi produk. Selain itu, nilai *Frame Per Second* (FPS) dan *inference time* pada saat model dijalankan akan menjadi acuan seberapa bagus model dalam melakukan deteksi produk pada saat diimplementasikan ke skenario dunia nyata.

Pengujian kedua dilakukan dengan memberikan jenis *dataset* yang berbeda untuk mengetahui pengaruh jenis *dataset* yang berisi gambar produk diam dengan *dataset* yang berisi gambar produk bergerak. *Dataset* dengan produk bergerak akan memiliki *motion blur* natural yang diharapkan dapat memberi gambaran produk pada skenario dunia nyata yang bergerak masuk ataupun keluar dari keranjang belanja. Seluruh *dataset* yang digunakan berisi gambar yang diambil pada kondisi pencahayaan yang cukup sehingga dapat memberikan detail yang lebih bagus pada setiap gambarnya.

Pengujian ketiga dilakukan dengan kondisi pencahayaan yang berbeda untuk mengetahui pengaruh pencahayaan terhadap hasil deteksi yang dilakukan. Kondisi pencahayaan pada setiap tempat dan waktu akan selalu berbeda, maka pada penelitian kali ini diberikan tambahan LED Strip yang diletakkan di sekeliling keranjang yang diharapkan dapat membantu supaya intensitas cahaya selalu cukup untuk melakukan deteksi. Pengujian akan dilakukan menggunakan beberapa *dataset* testing yang diambil pada kondisi cahaya gelap dan terang serta kondisi LED Strip menyala dan mati.

Pengujian keempat dilakukan pada keseluruhan sistem dengan kecepatan produk yang berbeda saat memasukkan dan mengeluarkan produk untuk mengetahui hasil dari keseluruhan sistem untuk mendeteksi produk yang masuk dan keluar dari keranjang. Pengujian akan dijalankan pada Raspberry Pi 4B mengetahui performa model saat sudah implementasi dan dijalankan di komputer pengujian untuk mengetahui seberapa besar performa sesungguhnya dari model yang telah dibuat. Pengujian akan dilakukan di ruangan yang memiliki cukup cahaya dengan intensitas cahaya sebesar 112 lux. Data yang akan diambil dan dianalisis adalah tingkat keberhasilan sistem dalam melakukan deteksi dari 5 produk yang akan dimasukkan dan dikeluarkan masing-masing sebanyak lima kali. Pengujian juga akan dilakukan dengan beberapa variasi kecepatan dalam memasukkan dan mengeluarkan produk. Variasi kecepatan yang digunakan adalah pelan dengan kecepatan sekitar 7 cm/s, sedang dengan kecepatan sekitar 36 cm/s, dan cepat dengan cara dijatuhkan yang menghasilkan kecepatan sekitar 124 cm/s. Nilai kecepatan tersebut didapat dari melihat perubahan titik produk pada penggaris yang diletakkan pada bagian belakang keranjang seperti pada Gambar 4.4.

Gambar 4.4 Peletakan penggaris untuk mengukur kecepatan

Pengujian terakhir dilakukan dengan menambahkan *sampling rate* untuk mengetahui pengaruh sistem ketika dibuat untuk dapat melakukan pengambilan beberapa *frame* gambar terlebih dahulu sebelum melakukan deteksi. Pengujian ini

bertujuan untuk mendapatkan hasil apakah penambahan *sampling rate* dapat meningkatkan akurasi dan kecepatan pergerakan produk atau tidak. Untuk melakukan pengujian perlu dilakukan pengubahan sistem supaya melakukan pengambilan beberapa *frame* gambar sebanyak besarnya *sampling rate* terlebih dahulu baru kemudian setiap *frame* tersebut akan dideteksi oleh sistem sesuai dengan urutannya. Karena sistem akan melakukan deteksi beberapa *frame* dan menyebabkan *delay* sistem yang lebih lama, maka skema penggunaan sistem juga perlu diubah menjadi menekan tombol setiap ingin memulai sistem pengambilan dan deteksi gambar. Namun hal tersebut menjadi kekurangan karena akan membuat sistem tidak dapat diganggu pada saat melakukan deteksi produk dan harus menunggu hingga seluruh proses deteksi yang sedang berjalan selesai untuk bisa melakukan deteksi produk selanjutnya.

Produk yang dideteksi pada pengujian empat dan lima akan menggunakan sisi depan dari produk saja untuk mendapatkan hasil yang lebih baik karena memiliki detail spesifik paling banyak. Hal tersebut juga dilakukan karena pada percobaan empat dan lima dilakukan untuk mengetahui pengaruh faktor yang tidak akan terpengaruh pada sisi dalam mendeteksi produk.

4.5. Pengujian 1: Penggunaan Variasi *Backbone* pada YOLOv5

Pada pengujian ini akan dilakukan perbandingan *metrics* yang didapat dari hasil pelatihan model YOLOv5 dengan dua buah variasi *backbone*, yaitu *original* dan *Swin Transformer*. Pengujian ini akan menghasilkan beberapa *metrics* yang akan menjadi acuan untuk menentukan seberapa baik *training* yang sudah dilakukan. *Metrics* yang akan diukur adalah *F1-Score*, mAP, dan *loss*. *Metrics loss* yang akan diukur yaitu *box loss*, *object loss*, dan *class loss*. Saat melakukan pengujian pada model, terdapat beberapa aspek yang dibuat tetap yaitu spesifikasi komputer pelatihan, *dataset* menggunakan *dataset* statis, *optimizer* menggunakan SGD, serta *epoch* yang berjumlah 100. Sedangkan untuk parameter yang akan diubah menyesuaikan jenis *backbone*-nya adalah resolusi *dataset* dan *batch size*. Pada *backbone original* resolusi *dataset* yang digunakan adalah 640x640 dan *batch size* yang digunakan adalah 128. Pada *backbone swin transformer* resolusi *dataset* yang digunakan adalah 224x224 dan *batch size* yang digunakan adalah 20.

Tabel 4.2 *Metrics* hasil terakhir pelatihan YOLOv5 menggunakan variasi *backbone*

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**	***Box Loss***	***Object Loss***	***Class Loss***
YOLOv5 dengan *backbone original*						
Train 1	98.70%	99.46%	96.93%	1.024%	0.418%	0.389%
Train 2	98.75%	99.50%	97.36%	1.006%	0.425%	0.398%
Train 3	98.28%	99.48%	97.30%	0.993%	0.421%	0.399%
Train 4	98.74%	99.49%	97.19%	1.038%	0.418%	0.369%
Train 5	98.70%	99.46%	96.93%	1.024%	0.418%	0.389%
Rata-rata	**98.64%**	**99.48%**	**97.14%**	**1.017%**	**0.420%**	**0.389%**
Standar Deviasi	**0.00201**	**0.00017**	**0.00202**	**0.00018**	**0.00003**	**0.00012**
YOLOv5 dengan *backbone swin transformer*						
Train 1	97.23%	98.98%	79.91%	2.089%	0.669%	0.782%
Train 2	96.71%	99.19%	79.72%	2.147%	0.694%	0.983%
Train 3	97.91%	99.15%	82.14%	2.126%	0.668%	0.864%
Train 4	98.21%	99.12%	82.37%	2.159%	0.672%	0.868%
Train 5	97.51%	99.19%	82.27%	2.114%	0.669%	0.873%
Rata-rata	**97.51%**	**99.13%**	**81.28%**	**2.127%**	**0.674%**	**0.874%**
Standar Deviasi	**0.00585**	**0.00087**	**0.01343**	**0.00028**	**0.00011**	**0.00071**

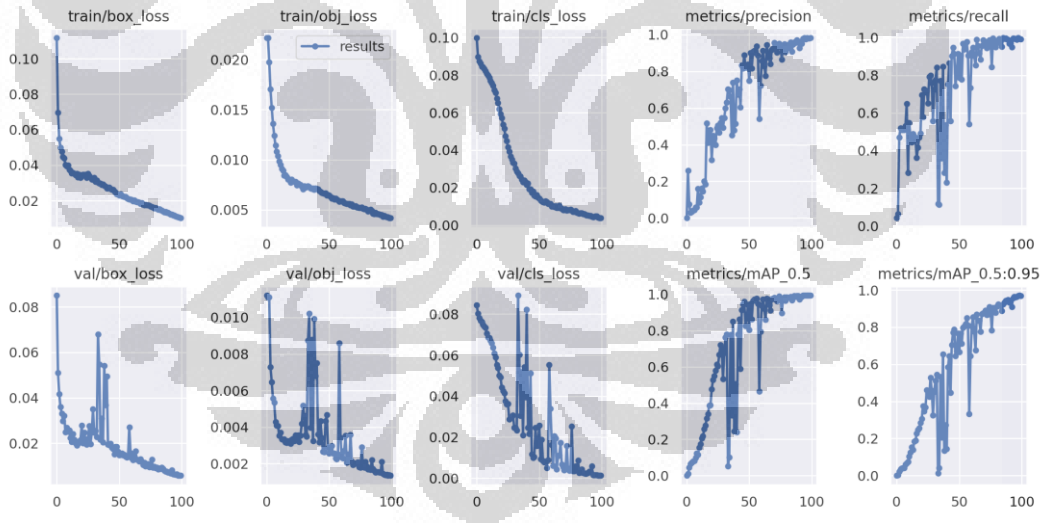
*nilai sudah dibulatkan.

Tabel 4.3 *Metrics* hasil terakhir validasi YOLOv5 menggunakan variasi *backbone*

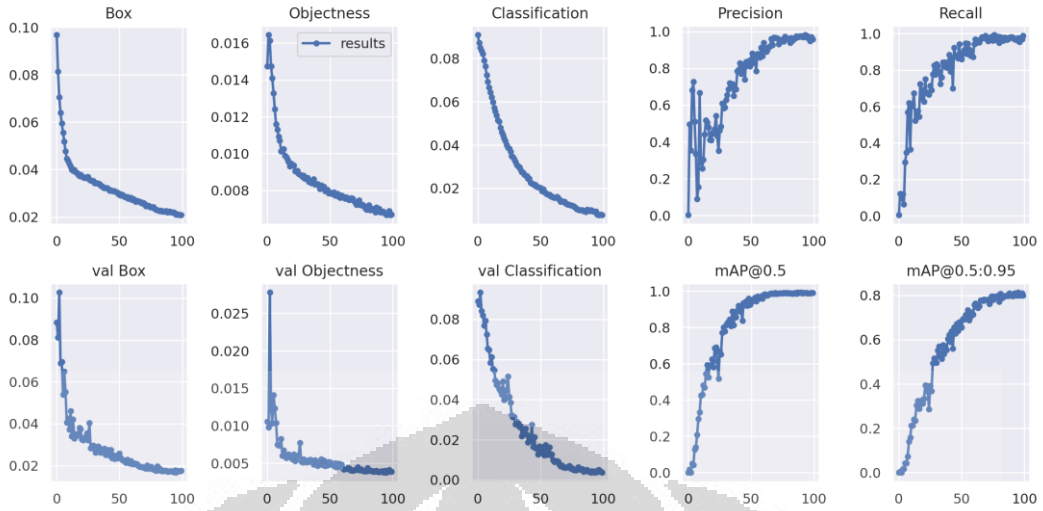
	YOLOv5 dengan *backbone original*			**YOLOv5 dengan *backbone swin transformer***		
	Box Loss	***Object Loss***	***Class Loss***	***Box Loss***	***Object Loss***	***Class Loss***
Valid 1	0.584%	0.135%	0.125%	1.751%	0.389%	0.369%
Valid 2	0.561%	0.129%	0.123%	1.749%	0.399%	0.533%
Valid 3	0.565%	0.131%	0.124%	1.688%	0.382%	0.420%
Valid 4	0.594%	0.134%	0.122%	1.631%	0.362%	0.343%
Valid 5	0.584%	0.135%	0.125%	1.690%	0.385%	0.413%
Rata-rata	**0.578%**	**0.133%**	**0.124%**	**1.702%**	**0.383%**	**0.416%**
Standar Deviasi	**0.00014**	**0.00003**	**0.00002**	**0.00050**	**0.00014**	**0.00073**

*nilai sudah dibulatkan.

Berdasarkan hasil percobaan dari variasi jenis *backbone* pada YOLOv5, dapat terlihat pengaruh perbedaan *backbone* terhadap akurasi dan *loss* model yang dihasilkan. Pada Tabel 4.2 dapat dilihat hasil pelatihan dari model YOLOv5 pada *epoch* terakhir menunjukkan penggunaan *backbone original* menghasilkan rata-rata yang lebih baik dibandingkan dengan *backbone swin transformer*. Hal tersebut bisa terjadi karena model yang berbasis CNN tidak memerlukan *dataset* dengan jumlah yang cukup besar untuk melakukan generalisasi berkat adanya *inductive biases* yang lebih kuat [30]. Dapat dilihat juga F1-Score dan mAP yang dihasilkan akan mempengaruhi nilai *loss* baik untuk pelatihan ataupun validasi. Pada Tabel 4.2 dan Tabel 4.3 terdapat nilai *box loss*, *object loss*, dan *class loss* yang dihasilkan dari proses validasi dan pelatihan yang dilakukan. Pada penggunaan *backbone original* dihasilkan nilai *loss* yang lebih rendah dibandingkan dengan *backbone swin transformer*. Maka berdasarkan nilai akurasi dan *loss* di percobaan ini didapatkan bahwa hasil jenis *backbone original* YOLOv5 lebih baik dari jenis *backbone swin transformer*.



Gambar 4.5 Grafik *metrics* selama pelatihan YOLOv5 dengan *backbone original*



Gambar 4.6 Grafik *metrics* selama pelatihan YOLOv5 dengan *backbone swin transformer*

Seperti yang telah dijelaskan sebelumnya mengenai model berbasis CNN yang dapat belajar lebih cepat, bisa dilihat pada Gambar 4.5 penggunaan *backbone original* membuat beberapa *metrics* yang dihasilkan seperti *validation loss*, *precision*, *recall*, dan mAP tidak diperlakukan dengan baik. Pada Gambar 4.5 di rentang *epoch* 30 sampai 70 terjadi peningkatan *validation loss* dan penurunan akurasi pada beberapa *epoch*, sedangkan pada Gambar 4.6 *validation loss* dan akurasi dapat meningkat menjadi lebih baik untuk setiap *epoch*-nya. Hal tersebut berpotensi menyebabkan *overfitting* pada *backbone original* karena nilai *training loss* yang dihasilkan selalu menurun dengan baik untuk setiap *epoch*-nya namun tidak dengan *validation loss*. Penurunan *training loss* tersebut menunjukkan model telah melakukan pelatihan dengan baik pada *dataset training* yang diberikan namun tidak berbanding lurus dengan *dataset validation*. Hal tersebut terjadi karena kuatnya *inductive bias* yang membuat model CNN akan berfokus untuk melakukan prediksi pada lokasi yang spesifik dalam suatu gambar saat melakukan pelatihan dan melakukan generalisasi berdasarkan data pada lokasi tersebut [31].

Tabel 4.4 *Metrics* hasil terbaik testing YOLOv5 menggunakan variasi *backbone*

	YOLOv5 dengan *backbone original*			**YOLOv5 dengan *backbone swin transformer***		
	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**	***F1-Score***	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	90.30%	70.80%	79.37%	94.80%	77.00%	84.98%
Test 2	93.80%	67.70%	78.64%	98.10%	76.60%	86.03%
Test 3	96.00%	66.50%	78.57%	94.90%	78.00%	85.62%
Test 4	92.30%	70.40%	79.88%	97.70%	75.50%	85.18%
Test 5	90.30%	70.80%	79.37%	95.40%	77.70%	85.65%
Rata-rata	**92.54%**	**69.24%**	**79.17%**	**96.18%**	**76.96%**	**85.49%**
Standar Deviasi	**0.02432**	**0.02006**	**0.00551**	**0.01593**	**0.00986**	**0.00415**

*nilai sudah dibulatkan.

Setelah melakukan pelatihan model dengan variasi *backbone* maka model harus dilakukan testing menggunakan beberapa data gabungan dari *dataset* statis dengan *dataset* bergerak. Hal tersebut bertujuan untuk mengetahui seberapa besar tingkat *overfitting* yang terjadi pada model dengan memberikan gambar yang belum pernah dilihatnya. Berdasarkan Tabel 4.4 dapat dilihat bahwa hasil testing model dengan *backbone swin transformer* memberikan nilai yang lebih baik dibandingkan dengan model yang menggunakan *backbone original*. Hal tersebut menunjukkan bahwa pernyataan sebelumnya benar terbukti karena terjadi *overfitting* yang lebih tinggi pada model dengan *backbone original* dibandingkan dengan model dengan *backbone swin transformer*. Maka berdasarkan nilai akurasi dari testing menggunakan *dataset* yang belum pernah dilihat oleh model didapatkan hasil bahwa jenis *backbone swin transformer* lebih baik dalam melakukan generalisasi objek sehingga bisa berhasil melakukan deteksi beberapa objek dari gambar yang belum pernah dilihatnya.

Tabel 4.5 Waktu dan ukuran hasil pelatihan YOLOv5 menggunakan variasi *backbone*

	YOLOv5 dengan *backbone original*		**YOLOv5 dengan *backbone swin transformer***	
	Lama Pelatihan (menit/*epoch*)	**Ukuran TFLite (MB)**	**Lama Pelatihan (menit/*epoch*)**	**Ukuran TFLite (MB)**
Test 1	4.598	7.7	9.360	155.4
Test 2	4.755	7.7	9.720	155.4
Test 3	4.845	7.7	8.024	155.4
Test 4	4.770	7.7	7.889	155.4
Test 5	4.841	7.7	8.400	155.4
Rata-rata	**4.762**	**7.7**	**8.679**	**155.4**
Standar Deviasi	**0.100**	**0.0**	**0.818**	**0.0**

*nilai sudah dibulatkan.

Terdapat beberapa variabel yang dapat menjadi parameter penentu dalam pemilihan model yaitu lama waktu pelatihan dan ukuran model yang dihasilkan. Berdasarkan Tabel 4.5 dari proses pelatihan yang dilakukan menggunakan variasi *backbone* pada model YOLOv5 menunjukkan bahwa waktu pelatihan yang dibutuhkan oleh *backbone swin transformer* lebih lama hampir dua kali dari waktu pelatihan yang dibutuhkan oleh *backbone original*. Hal tersebut terjadi karena adanya perbedaan *batch size* yang digunakan pada saat melakukan pelatihan di komputer dengan spesifikasi yang sama. *Batch size* yang digunakan berbeda karena pada *model* berbasis *transformer* memerlukan *resource* komputasi terutama pada memori dengan ukuran yang lebih besar untuk dapat melakukan mekanisme *self-attention.* Mekanisme *self-attention* pada model berbasis *transformer* memerlukan *resource* yang besar karena harus memperhatikan detail informasi yang lebih banyak dibandingkan dengan *attention* pada model berbasis CNN [32]. Karena adanya mekanisme *attention* yang dirancang khusus maka diperlukan banyak parameter pada *backbone swin transformer* yang menyebabkan *output* yang dihasilkan juga memiliki ukuran yang lebih besar dibandingkan *backbone original* yang hanya menggunakan CNN. Berdasarkan Tabel 4.5 dapat dilihat ukuran model TFLite yang dihasilkan oleh model YOLOv5 dengan *backbone swin transformer* memiliki ukuran 155.4 MB sedangkan pada model YOLOv5 dengan *backbone original*

hanya memiliki ukuran sebesar 7.7 MB. Maka berdasarkan lama waktu pelatihan dan ukuran model didapatkan hasil bahwa jenis *backbone original* lebih cepat untuk melakukan pelatihan pada *resource* yang sama dan memiliki *output* model dengan ukuran yang lebih kecil.

Tabel 4.6 Hasil testing YOLOv5 dengan variasi *backbone*

	Komputer Pengujian		**Raspberry Pi 4B**	
	***Inference Time* (detik)**	***Frame per Seconds***	***Inference Time* (detik)**	***Frame per Seconds***
YOLOv5 dengan *backbone original*				
Test 1	0.234	3.974	0.586	1.306
Test 2	0.240	4.010	0.865	0.601
Test 3	0.253	3.759	1.122	0.293
Test 4	0.234	3.776	1.317	0.282
Test 5	0.230	3.841	0.656	1.216
Rata-rata	**0.238**	**3.872**	**0.909**	**0.740**
Standar Deviasi	**0.009**	**0.114**	**0.309**	**0.494**
YOLOv5 dengan *backbone swin transformer*				
Test 1	5.142	0.165	99.973	0.0198
Test 2	6.823	0.154	98.159	0.0101
Test 3	6.432	0.126	107.456	0.0101
Test 4	5.732	0.180	96.184	0.0104
Test 5	5.654	0.172	92.956	0.0190
Rata-rata	**5.957**	**0.159**	**98.946**	**0.014**
Standar Deviasi	**0.668**	**0.021**	**5.422**	**0.005**

*nilai sudah dibulatkan.

Pada penelitian ini model yang telah dibuat akan diimplementasikan pada Raspberry Pi 4B untuk selanjutnya diletakkan prototipe pada *smart trolley* yang telah dibuat. Maka untuk mendapatkan hasil model yang paling cocok dengan kebutuhan diperlukan pengujian model yang dijalankan langsung pada perangkat Raspberry Pi 4B. Namun sebagai perbandingan model juga akan dijalankan pada komputer pengujian sebagai perangkat yang cukup baik dalam menjalankan model. Berdasarkan Tabel 4.6 dapat dilihat *backbone original* dapat melakukan deteksi

pada gambar dengan lebih cepat dibandingkan *backbone swin transformer*. Hal tersebut sangat berkaitan dengan hasil sebelumnya yang menunjukkan ukuran model *backbone original* jauh lebih kecil dibandingkan model *swin transformer* sehingga lebih ringan untuk dijalankan.

Karena pada penelitian kali ini model yang dicari harus dapat melakukan deteksi secara *real-time* pada perangkat Raspberry Pi 4B, maka diperlukan model yang dapat memiliki waktu *inference* yang lebih kecil namun tetap memiliki akurasi yang tinggi dan *loss* yang rendah. Maka berdasarkan kebutuhan dan hasil pengujian pertama dipilih model YOLOv5 dengan *backbone original* untuk digunakan pada pengujian selanjutnya.

4.6. Pengujian 2: Variasi *Dataset* dengan Gambar dari *Frame* Video

Pada pengujian ini akan dilakukan perbandingan *metrics* yang didapat dari hasil pelatihan model YOLOv5 dengan menggunakan beberapa variasi jenis *dataset*. Untuk variasi jenis *dataset* yang akan digunakan adalah *dataset* statis, *dataset* statis dengan *augmentasi* blur, *dataset* bergerak, dan kombinasi *dataset* bergerak dengan *dataset* statis yang menggunakan *augmentasi* blur. Pengujian ini akan menghasilkan beberapa *metrics* yang akan menjadi acuan untuk menentukan seberapa baik *training* yang sudah dilakukan. *Metrics* yang akan diukur adalah *Precision, Recall, F1-Score*, dan mAP. Terdapat juga *metrics loss* yang akan diukur yaitu *box loss, object loss,* dan *class loss*. Saat melakukan pengujian pada model, terdapat beberapa aspek yang dibuat tetap yaitu spesifikasi komputer pelatihan, *backbone model* menggunakan yang *original, optimizer* menggunakan SGD, serta *epoch* yang berjumlah 100.

Tabel 4.7 *Metrics* hasil *epoch* terakhir pelatihan *dataset* statis

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**	***Box Loss***	***Object Loss***	***Class Loss***
Train 1	96.71%	96.38%	93.10%	1.166%	0.478%	0.447%
Train 2	97.32%	98.41%	95.21%	1.113%	0.449%	0.453%
Train 3	97.32%	98.41%	95.21%	1.113%	0.449%	0.453%
Train 4	97.32%	98.41%	95.21%	1.113%	0.449%	0.453%
Train 5	97.32%	98.41%	95.21%	1.113%	0.449%	0.453%
Rata-rata	**97.20%**	**98.00%**	**94.79%**	**1.123%**	**0.455%**	**0.452%**
Standar Deviasi	**0.00273**	**0.00909**	**0.00940**	**0.00024**	**0.00013**	**0.00003**

*nilai sudah dibulatkan.

Tabel 4.8 *Metrics* hasil *epoch* terakhir pelatihan *dataset* statis *blur*

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**	***Box Loss***	***Object Loss***	***Class Loss***
Train 1	99.87%	99.50%	97.19%	0.949%	0.387%	0.173%
Train 2	99.84%	99.50%	97.41%	0.931%	0.387%	0.185%
Train 3	99.43%	99.50%	97.31%	0.956%	0.389%	0.204%
Train 4	99.75%	99.50%	97.61%	0.959%	0.389%	0.185%
Train 5	99.41%	99.50%	97.28%	0.939%	0.388%	0.185%
Rata-rata	**99.66%**	**99.50%**	**97.36%**	**0.947%**	**0.388%**	**0.186%**
Standar Deviasi	**0.00222**	**0.00000**	**0.00162**	**0.00012**	**0.00001**	**0.00011**

*nilai sudah dibulatkan.

Tabel 4.9 *Metrics* hasil *epoch* terakhir pelatihan *dataset* bergerak

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**	***Box Loss***	***Object Loss***	***Class Loss***
Train 1	87.07%	91.17%	81.35%	1.948%	0.756%	2.906%
Train 2	87.07%	91.17%	81.35%	1.948%	0.756%	2.906%
Train 3	87.07%	91.17%	81.35%	1.948%	0.756%	2.906%
Train 4	87.07%	91.17%	81.35%	1.948%	0.756%	2.906%
Train 5	87.07%	91.17%	81.35%	1.948%	0.756%	2.906%
Rata-rata	**87.07%**	**91.17%**	**81.35%**	**1.948%**	**0.756%**	**2.906%**
Standar Deviasi	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**

*nilai sudah dibulatkan.

Tabel 4.10 *Metrics* hasil *epoch* terakhir pelatihan kombinasi *dataset* bergerak dengan statis *blur*

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**	***Box Loss***	***Object Loss***	***Class Loss***
Train 1	99.65%	99.50%	96.26%	1.040%	0.454%	0.301%
Train 2	99.45%	99.49%	96.14%	1.026%	0.459%	0.316%
Train 3	99.48%	99.49%	96.13%	1.020%	0.443%	0.329%
Train 4	99.34%	99.50%	96.65%	1.044%	0.444%	0.328%
Train 5	99.72%	99.50%	96.12%	1.016%	0.449%	0.315%
Rata-rata	**99.53%**	**99.50%**	**96.26%**	**1.029%**	**0.450%**	**0.318%**
Standar Deviasi	**0.00153**	**0.00003**	**0.00226**	**0.00013**	**0.00007**	**0.00011**

*nilai sudah dibulatkan.

Berdasarkan hasil pelatihan model pada beberapa variasi *dataset* memberikan hasil bahwa *dataset* bergerak memiliki akurasi yang paling rendah dengan rata-rata nilai *F1-Score* 87.07%, mAP pada *threshold* 0.5 91.17%, dan mAP pada *threshold* 0.5 sampai 0.95 81.35%. Karena akurasi yang dihasilkan dari variasi *dataset* bergerak rendah maka hal tersebut akan mempengaruhi nilai *loss* yang dihasilkan, sehingga variasi *dataset* bergerak memiliki rata-rata *loss* yang paling besar juga. Hal tersebut bisa terjadi karena pada *dataset* bergerak gambar yang digunakan adalah gambar bergerak yang berasal dari *frame* video sehingga tidak terdapat gambar statis yang memberikan gambaran detail dari tiap produk. Untuk variasi *dataset* yang terbaik terdapat pada variasi *dataset* statis *blur* dan gabungan *dataset* bergerak dengan statis *blur* dengan selisih nilai yang kecil. Hal tersebut terjadi karena pada *dataset* statis *blur* memiliki variasi statis *blur* yang membuat model bisa mengenali lebih banyak jenis produk, untuk kombinasi dengan variasi bergerak memiliki nilai yang mirip dengan variasi statis *blur* karena pada variasi ini hanya melakukan penambahan data *blur* yang berasal dari *dataset* bergerak.

Tabel 4.11 *Metrics* hasil terbaik validasi dan testing *dataset* statis

	Validasi			**Testing**		
	Box Loss	***Object Loss***	***Class Loss***	***F1-Score***	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	0,644%	0,141%	0,173%	80,16%	79,20%	72,10%
Test 2	0,634%	0,144%	0,169%	78,55%	78,60%	72,40%
Test 3	0,634%	0,144%	0,169%	78,55%	78,60%	72,40%
Test 4	0,634%	0,144%	0,169%	78,55%	78,60%	72,40%
Test 5	0,634%	0,144%	0,169%	78,55%	78,60%	72,40%
Rata-rata	**0,636%**	**0,143%**	**0,170%**	**78,87%**	**78,72%**	**72,34%**
Standar Deviasi	**0,00005**	**0,00001**	**0,00002**	**0,00721**	**0,00268**	**0,00134**

*nilai sudah dibulatkan.

Tabel 4.12 *Metrics* hasil terbaik validasi dan testing *dataset* statis *blur*

	Validasi			**Testing**		
	Box Loss	***Object Loss***	***Class Loss***	***F1-Score***	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	0,589%	0,132%	0,045%	90,66%	88,90%	84,80%
Test 2	0,570%	0,131%	0,044%	91,28%	89,60%	85,40%
Test 3	0,610%	0,130%	0,060%	90,39%	89,50%	85,40%
Test 4	0,598%	0,132%	0,046%	90,78%	89,20%	85,10%
Test 5	0,599%	0,130%	0,068%	90,61%	87,70%	84,20%
Rata-rata	**0,593%**	**0,131%**	**0,053%**	**90,74%**	**88,98%**	**84,98%**
Standar Deviasi	**0,00015**	**0,00001**	**0,00011**	**0,00333**	**0,00766**	**0,00502**

*nilai sudah dibulatkan.

Tabel 4.13 *Metrics* hasil terbaik validasi dan testing *dataset* bergerak

	Validasi			**Testing**		
	Box Loss	***Object Loss***	***Class Loss***	***F1-Score***	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	0,683%	0,214%	1,094%	37,50%	29,40%	21,90%
Test 2	0,683%	0,214%	1,094%	37,50%	29,40%	21,90%
Test 3	0,683%	0,214%	1,094%	37,50%	29,40%	21,90%
Test 4	0,683%	0,214%	1,094%	37,50%	29,40%	21,90%
Test 5	0,683%	0,214%	1,094%	37,50%	29,40%	21,90%
Rata-rata	**0,683%**	**0,214%**	**1,094%**	**37,50%**	**29,40%**	**21,90%**
Standar Deviasi	**0,00000**	**0,00000**	**0,00000**	**0,00000**	**0,00000**	**0,00000**

*nilai sudah dibulatkan.

Tabel 4.14 *Metrics* hasil terbaik validasi dan testing kombinasi *dataset* bergerak dengan statis *blur*

	Validasi			**Testing**		
	Box Loss	***Object Loss***	***Class Loss***	***F1-Score***	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	0,567%	0,135%	0,049%	99,60%	99,50%	96,50%
Test 2	0,569%	0,139%	0,048%	99,15%	99,50%	96,60%
Test 3	0,588%	0,140%	0,046%	99,45%	99,50%	96,50%
Test 4	0,564%	0,133%	0,059%	99,30%	99,50%	96,80%
Test 5	0,569%	0,138%	0,039%	99,70%	99,50%	96,40%
Rata-rata	**0,571%**	**0,137%**	**0,048%**	**99,44%**	**99,50%**	**96,56%**
Standar Deviasi	**0,00010**	**0,00003**	**0,00007**	**0,00222**	**0,00000**	**0,00152**

*nilai sudah dibulatkan.

Selain model dilatih, model juga harus divalidasi pada saat proses pelatihan serta dites setelah selesai melakukan pelatihan. Pada hasil validasi yang dilakukan didapatkan hasil yang sama seperti proses pelatihan yaitu *dataset* bergerak menjadi variasi *dataset* yang memiliki nilai *loss* tertinggi. Untuk nilai *loss* terendah juga didapatkan hasil yang sama dengan proses pelatihan yaitu *dataset* statis *blur* dan *dataset* gabungan *dataset* bergerak dengan statis *blur* dengan selisih nilai yang sedikit. Pada hasil testing yang di lakukan *dataset* bergerak memiliki nilai akurasi

paling rendah karena pada *dataset* ini tidak terdapat gambar yang memperlihatkan detail produk dan pada proses testing *dataset* yang digunakan adalah kombinasi *dataset* bergerak dengan statis. Untuk nilai akurasi yang paling tinggi didapatkan pada variasi kombinasi *dataset* bergerak dengan statis *blur* karena pada jenis *dataset* ini telah terdapat berbagai macam jenis gambar sehingga dapat dengan mudah mengenali gambar yang diberikan pada proses testing.

Setelah dilakukan pengujian didapat bahwa variasi *dataset* statis *blur* memiliki nilai akurasi yang paling tinggi pada fase *training* dan *loss* yang paling rendah pada fase *training* dan validasi. Namun nilai akurasi yang didapatkan variasi *dataset* statis *blur* pada fase testing tidak sebaik variasi kombinasi *dataset* bergerak dengan statis *blur* dengan selisih nilai F1-Score sebesar 8.7%, mAP dengan *threshold* 0.5 sebesar 10.52%, dan mAP dengan *threshold* 0.5 sampai 0.95 sebesar 11.58%. Karena hal tersebut maka variasi yang dipilih berdasarkan hasil pengujian kedua ini adalah variasi kombinasi *dataset* bergerak dan statis *blur* karena memiliki akurasi testing terbaik serta memiliki akurasi dan *loss* yang tidak berbeda jauh dengan variasi statis *blur* pada fase *training* dan validasi.

4.7. Pengujian 3: Variasi Intensitas Cahaya

Pada pengujian ini akan dilakukan perbandingan deteksi yang dilakukan pada kondisi dengan intensitas cahaya yang berbeda-beda. Pada pengujian ini terdapat tiga kondisi pencahayaan yang digunakan untuk memberikan hasil pengaruh dari intensitas cahaya terhadap hasil deteksi yang dilakukan. Kondisi pertama yaitu kondisi dengan cahaya matahari yang redup dengan intensitas cahaya sebesar 100 lux. Kondisi kedua yaitu kondisi pencahayaan yang hanya berasal dari lampu ruangan yang memiliki intensitas cahaya 40 lux. Kondisi ketiga yaitu kondisi pencahayaan yang bersumber dari lampu ruangan dan lampu yang ada di keranjang dengan intensitas cahaya 80 lux. *Metrics* yang akan diukur pada pengujian kali ini adalah *F1-Score* dan mAP untuk mengetahui akurasi yang dihasilkan.

Tabel 4.15 *Metrics* hasil testing YOLOv5 dengan intensitas cahaya 100 lux

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	34,37%	25,20%	12,20%
Test 2	34,53%	30,00%	13,60%
Test 3	35,68%	30,50%	12,90%
Test 4	34,48%	28,90%	14,80%
Test 5	33,44%	28,80%	13,40%
Rata-rata	**34,50%**	**28,68%**	**13,38%**
Standar Deviasi	**0,00796**	**0,02075**	**0,00960**

*nilai sudah dibulatkan.

Tabel 4.16 *Metrics* hasil testing YOLOv5 dengan intensitas cahaya 40 lux

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	4,64%	1,68%	0,53%
Test 2	12,54%	3,18%	0,76%
Test 3	18,89%	10,10%	4,23%
Test 4	22,42%	12,50%	4,01%
Test 5	13,56%	8,38%	2,55%
Rata-rata	**14,41%**	**7,17%**	**2,41%**
Standar Deviasi	**0,06778**	**0,04597**	**0,01745**

*nilai sudah dibulatkan.

Tabel 4.17 *Metrics* hasil testing YOLOv5 dengan intensitas cahaya 80 lux

	F1-Score	**mAP_0.5**	**mAP_0.5:0.95**
Test 1	26,75%	20,80%	9,02%
Test 2	24,81%	21,70%	10,60%
Test 3	27,31%	23,30%	11,50%
Test 4	23,22%	22,50%	11,30%
Test 5	23,51%	20,60%	10,10%
Rata-rata	**25,12%**	**21,78%**	**10,50%**
Standar Deviasi	**0,01854**	**0,01139**	**0,01000**

*nilai sudah dibulatkan.

Pada pengujian kali ini dilakukan untuk mendapatkan hasil yang lebih menggambarkan keadaan nyata dengan berbagai intensitas cahaya. Berdasarkan hasil percobaan didapatkan akurasi berdasarkan F1-Score yang cukup buruk jika dibandingkan dengan fase pelatihan sebelumnya. Hal tersebut terjadi disebabkan oleh faktor spesifikasi kamera yang kurang baik karena kamera yang digunakan memiliki spesifikasi 5-megapixel dan dikoneksikan ke Raspberry Pi 4B sehingga gambar yang diambil memiliki detail gambar yang kurang baik dan terdapat banyak *noise* yang mengganggu di dalam gambar. Selain itu total *frame* yang dapat ditangkap oleh kamera dengan Raspberry Pi 4B memiliki jumlah total kurang dari 30 *frame* dalam waktu satu detik yang membuat gambar menjadi sangat *blur* dan objek sangat sulit untuk terdeteksi. Untuk hasil pengujian pengaruh intensitas cahaya didapatkan hasil bahwa semakin besar intensitas cahaya akan sangat berpengaruh terhadap akurasi yang dihasilkan baik itu F1-Score ataupun mAP. Berdasarkan hasil pengujian, jika proses deteksi hanya dilakukan dengan cahaya lampu ruangan dengan intensitas cahaya sebesar 40 lux maka hasil yang didapatkan adalah akurasi F1-Score model adalah 14.41%, mAP dengan *threshold* 0.5 sebesar 7.17%, dan mAP dengan *threshold* 0.5 sampai 0.95 sebesar 2.41%. Jika dibandingkan dengan intensitas cahaya yang diberikan dari lampu di sekitar trolley maka akurasi yang didapatkan adalah F1-Score sebesar 25.12%, mAP dengan *threshold* 0.5 sebesar 21.78%, dan mAP dengan *threshold* 0.5 sampai 0.95 sebesar 10.50%. Maka bisa dikatakan pemberian lampu di sekitar troli dapat meningkatkan akurasi F1-Score sebesar 1.74 kali, mAP dengan *threshold* 0.5 sebesar 3.04 kali, dan mAP dengan *threshold* 0.5 sampai 0.95 sebesar 4.36 kali. Hal tersebut membuktikan bahwa penggunaan lampu di sekitar troli dapat memberikan pengaruh yang cukup besar pada hasil akurasi terutama pada kondisi pencahayaan yang gelap.

4.8. Pengujian 4: Pengujian Sistem dengan Variasi Kecepatan Produk

Pada pengujian ini akan dilakukan perbandingan sistem dalam melakukan deteksi produk yang masuk dan keluar keranjang dengan kecepatan produk yang berbeda-beda. Pada pengujian ini terdapat tiga kecepatan produk pada saat bergerak untuk memberikan hasil performa dari keseluruhan sistem dalam melakukan deteksi

produk. Kecepatan produk yang digunakan pada percobaan ini adalah pelan dengan kecepatan sekitar 7 cm/s, sedang dengan kecepatan sekitar 36 cm/s, dan cepat dengan cara dijatuhkan yang menghasilkan kecepatan sekitar 124 cm/s. Hasil *recognition rate* produk didapat dari uji coba yang dilakukan sebanyak lima kali pengulangan untuk setiap produknya dan dilakukan secara terus-menerus.

Tabel 4.18 *Recognition rate* variasi kecepatan produk

	Pelan			**Sedang**			**Cepat**
	Masuk	**Keluar**	**Rata-rata**	**Masuk**	**Keluar**	**Rata-rata**	**Masuk**
Komputer Pengujian							
Chitato Ayam 68 gram	60%	100%	80%	80%	20%	50%	0%
Nice Tissue 250 sheets	20%	20%	20%	100%	20%	60%	0%
Mie Sedaap Goreng 90 gram	40%	40%	40%	40%	0%	20%	0%
Chitato Sapi 68 gram	100%	60%	80%	60%	40%	50%	0%
Ultra Milk Stroberi 250 ml	60%	80%	70%	40%	0%	20%	0%
Raspberry Pi 4B							
Chitato Ayam 68 gram	60%	40%	50%	0%	0%	0%	0%
Nice Tissue 250 sheets	20%	0%	10%	0%	0%	0%	0%
Mie Sedaap Goreng 90 gram	0%	0%	0%	0%	0%	0%	0%
Chitato Sapi 68 gram	60%	0%	30%	0%	0%	0%	0%
Ultra Milk Stroberi 250 ml	20%	20%	20%	0%	0%	0%	0%

Berdasarkan hasil pengujian sistem dengan variasi kecepatan produk memberikan hasil bahwa semakin lambat produk bergerak maka produk akan lebih mudah untuk dideteksi. Dapat dilihat bahwa hasil deteksi yang dilakukan pada perangkat komputer pengujian lebih baik dari Raspberry Pi 4B karena *resource* pemrosesan

yang dapat digunakan lebih besar sehingga gambar produk yang ditangkap bisa lebih banyak dan deteksi model bisa memberikan hasil yang lebih baik juga. Selain itu pemrosesan dalam pengambilan *frame* dari kamera pada komputer pengujian lebih baik daripada di Raspberry Pi 4B sehingga mempengaruhi hasil deteksi.

Berdasarkan hasil juga dapat dilihat bahwa Mie Sedaap Goreng 90 gram cukup sulit untuk dideteksi karena ukurannya yang tidak besar dan dibungkus oleh plastik yang membuat bentuknya bisa dengan mudah untuk berubah. Sedangkan produk Chitato Ayam 68 gram ataupun Chitato Sapi 68 gram memiliki rata-rata hasil deteksi yang lebih baik dari yang lainnya dikarenakan memiliki ukuran produk yang cukup lebar sehingga memberikan banyak detail yang dapat dilihat model dan memiliki ukuran yang paling tinggi sehingga memungkinkan produk untuk tertangkap di dalam *frame* lebih banyak dari yang lainnya. Untuk Nice Tissue 250 sheets memiliki rata-rata akurasi yang cukup rendah dikarenakan produk hanya dapat terdeteksi pada saat berada di bagian atas saja sehingga tidak dapat dikatakan masuk ke dalam keranjang. Hal tersebut terjadi karena tinggi yang dimiliki Nice Tissue 250 sheets kurang dari Chitato Ayam 68 gram, Chitato Sapi 68 gram, ataupun Ultra Milk Stroberi 250 ml. Namun ketika digerakkan dengan kecepatan sedang Nice Tissue 250 sheets mendapatkan nilai *recognition rate* yang paling tinggi karena dengan kecepatan yang lebih besar terdapat *blur* yang ada di sisi atas dan bawah produk yang membuat Nice Tissue 250 sheets saat terdeteksi memiliki tinggi yang lebih dari aslinya dan berhasil terdeteksi dengan lebih baik.

Untuk hasil deteksi produk dengan kecepatan sedang pada komputer pengujian, beberapa produk masih dapat terdeteksi dengan baik dikarenakan model yang digunakan telah dilatih dengan *dataset* yang memiliki variasi *blur* paling banyak sehingga model masih bisa mendeteksi produk walaupun sedang bergerak dengan kecepatan sedang. Produk yang memiliki kecepatan cepat tidak ada yang dapat dideteksi karena produk bergerak terlalu cepat dan sulit untuk mendapatkan gambar produknya. Hal tersebut membuat pada komputer pengujian hanya terdapat satu *frame* saja yang berhasil menangkap gambar produk saat sedang bergerak dengan kecepatan cepat, sedangkan pada Raspberry Pi 4B tidak dapat menangkap gambar produk sama sekali. Pada Raspberry Pi 4B juga dapat dilihat bahwa hasil deteksi

pada produk yang memiliki kecepatan sedang tidak ada yang berhasil terdeteksi karena sistem terlalu lama dalam mendeteksi satu *frame* sehingga total *frame* dengan produk yang dapat ditangkap hanya sedikit. Masalah-masalah yang telah dijelaskan pada pengujian tiga juga menjadi salah satu faktor yang menyebabkan Raspberry Pi 4B tidak dapat mendeteksi produk dengan kecepatan sedang dan kesulitan dalam mendeteksi produk dengan kecepatan pelan. Raspberry Pi 4B bisa dikatakan kurang *powerful* dalam melakukan pekerjaan ini karena pada komputer pengujian dengan model yang sama dapat melakukan deteksi dengan cukup baik.

Maka dari pengujian ini bisa dikatakan kecepatan produk yang lebih pelan bisa dideteksi dengan lebih baik karena terdapat lebih banyak *frame* dengan produk yang dapat dideteksi. Dari pengujian ini juga bisa dikatakan kecepatan ideal yang dapat digunakan pada saat proses deteksi di komputer pengujian adalah hingga 36 cm/s dan untuk proses yang dilakukan di Raspberry Pi 4B adalah di bawah 7 cm/s. Lalu produk yang memiliki ukuran dengan lebar yang cukup dan tidak terlalu kecil akan memberikan detail spesifik yang lebih pada saat deteksi dan meningkatkan hasil deteksi yang dilakukan menjadi lebih baik. Untuk produk yang memiliki ukuran tinggi yang kurang besar bisa menjadi kesulitan saat melakukan deteksi di kecepatan pelan namun akan menjadi normal pada kecepatan yang lebih tinggi karena terdapat *motion blur* yang akan membantu memberikan tambahan tinggi.

4.9. Pengujian 5: Penambahan *Sampling Rate*

Pada pengujian ini akan dilakukan perbandingan sistem dalam melakukan deteksi produk jika *sampling rate* yang digunakan diubah menjadi 100 untuk kecepatan yang sedang dan 50 untuk kecepatan yang cepat. Pada pengujian ini akan digunakan dua kecepatan produk pada saat bergerak untuk memberikan hasil performa dari keseluruhan sistem saat melakukan deteksi dengan *sampling rate* yang diubah. Kecepatan produk yang digunakan pada percobaan ini adalah sedang dengan kecepatan sekitar 36 cm/s dan cepat dengan cara dijatuhkan yang menghasilkan kecepatan sekitar 124 cm/s. Kecepatan pelan tidak ada karena *frame* yang perlu diambil di awal akan sangat banyak dan menimbulkan *delay* deteksi yang sangat lama. Hasil *recognition rate* produk didapat dari uji coba yang dilakukan sebanyak lima kali pengulangan untuk setiap produknya dan dilakukan secara terus-menerus.

Tabel 4.19 *Recognition rate* deteksi dengan metode penambahan *sampling rate*

	Sedang			**Cepat**
	Masuk	**Keluar**	**Rata-rata**	**Masuk**
Komputer Pengujian				
Chitato Ayam 68 gram	60%	60%	60%	20%
Nice Tissue 250 sheets	100%	100%	100%	60%
Mie Sedaap Goreng 90 gram	40%	40%	40%	0%
Chitato Sapi 68 gram	80%	60%	70%	20%
Ultra Milk Stroberi 250 ml	40%	60%	50%	0%
Raspberry Pi 4B				
Chitato Ayam 68 gram	0%	0%	0%	0%
Nice Tissue 250 sheets	0%	0%	0%	0%
Mie Sedaap Goreng 90 gram	0%	0%	0%	0%
Chitato Sapi 68 gram	20%	20%	20%	0%
Ultra Milk Stroberi 250 ml	20%	0%	10%	0%

*kecepatan pelan tidak ada karena memerlukan waktu deteksi yang sangat lama.

Berdasarkan hasil pengujian sistem dengan menambahkan *sampling rate* memberikan hasil bahwa ketika sistem dapat mendeteksi dengan menggunakan *sampling* gambar yang lebih banyak dan diambil secara langsung di awal tanpa *delay*, sistem akan lebih mudah dalam melakukan deteksi. Hal tersebut karena gambar yang diambil sekaligus di awal akan memberikan *frame* yang terdapat produk di dalamnya dengan jumlah yang lebih banyak sehingga sistem dapat melakukan deteksi yang lebih banyak juga pada produk. Pada saat menggunakan metode *sampling* gambar juga memberikan dampak keakuratan sistem deteksi yang lebih baik terutama dalam hal klasifikasi. Maka dari itu, jika dilihat hasil pada Tabel 4.19. memberikan nilai *recognition rate* yang lebih baik dari hasil percobaan empat yang ada pada Tabel 4.18. Namun untuk hasil pengujian pada Raspberry Pi 4B walaupun meningkat dari sebelumnya tetapi masih ada beberapa produk yang tidak dapat dideteksi faktor-faktor masalah yang ada pada Raspberry Pi 4B seperti pada penelitian sebelumnya.

Berdasarkan hasil juga dapat dilihat pada kecepatan sedang ketika menggunakan metode *sampling* gambar, rata-rata *recognition rate* hasil deteksi produk dengan

kecepatan sedang dari seluruh kelas produk pada komputer pengujian dapat meningkat dan produk Chitato Sapi 68 gram serta Ultra Milk Stroberi 250 ml dapat terdeteksi sesekali oleh Raspberry Pi 4B. Pada kecepatan cepat beberapa produk berhasil terdeteksi oleh sistem pada komputer pengujian karena *frame* yang terdapat produk bisa tertangkap lebih banyak ketika menggunakan metode *sampling* gambar. Walaupun untuk produk yang dapat terdeteksi pada kecepatan cepat hanya produk yang berukuran cukup lebar sehingga Mie Sedaap Goreng 90 gram dan Ultra Milk Stroberi 250 ml masih belum bisa terdeteksi. Untuk hasil deteksi terbaik pada kecepatan cepat adalah Nice Tissue 250 sheets karena produk memang sudah dapat dideteksi dengan baik namun pada pengujian empat masih sering tidak terdeteksi karena tingginya yang kurang, sehingga ketika terdapat *motion blur* di kecepatan yang lebih tinggi akan menambah tinggi dari produk yang terdeteksi juga.

Maka dari pengujian ini bisa dikatakan penambahan *sampling rate* saat melakukan deteksi dapat memberikan dampak yang lebih baik terhadap hasil deteksi. Dari pengujian ini juga bisa dikatakan saat melakukan penambahan *sampling rate*, kecepatan yang dapat digunakan pada saat proses deteksi yang dilakukan di komputer pengujian dapat meningkat hingga 124 cm/s pada produk-produk dengan ukuran yang cukup lebar sedangkan untuk proses deteksi yang dilakukan pada Raspberry Pi 4B masih sama berada di kisaran 7 cm/s.

4.10. Analisis Keseluruhan

Pengujian pertama dilakukan untuk mendapatkan hasil yang terbaik dari perbandingan model YOLOv5 yang menggunakan *backbone original* dengan *backbone swin transformer*. Hasil pengujian menunjukkan model YOLOv5 yang menggunakan *backbone original* memiliki potensi yang lebih besar untuk terjadinya *overfit* dibandingkan dengan *backbone swin transformer* karena hasil testing menunjukkan nilai akurasi dan *loss backbone swin transformer* yang lebih baik dibandingkan dengan *backbone original*. Model YOLOv5 dengan *backbone swin transformer* juga masih memiliki banyak potensi untuk dilatih dengan fitur yang lebih banyak sedangkan *backbone original* terlihat sudah berada di titik jenuhnya. Pada Model YOLOv5 dengan *backbone original* memiliki akurasi dan *loss* yang lebih baik dari *backbone swin transformer*. Selain itu YOLOv5 dengan

backbone original memiliki ukuran TFLite yang lebih kecil dan dapat berjalan lebih cepat dibandingkan dengan *backbone swin transformer*. Karena model yang dicari pada penelitian ini harus dapat melakukan deteksi secara *real-time* pada perangkat Raspberry Pi 4B, sehingga dipilih model YOLOv5 dengan *backbone original*.

Pengujian kedua dilakukan untuk mendapatkan hasil yang terbaik dari perbandingan variasi *dataset* yang digunakan. Hasil pengujian menunjukkan variasi *dataset* statis *blur* memiliki nilai akurasi paling tinggi pada fase *training* dan *loss* paling rendah pada fase *training* dan validasi. Namun dari hasil kesulurah fase didapatkan kombinasi *dataset* bergerak dengan statis *blur* memiliki hasil yang lebih baik dari *dataset* statis *blur* karena memiliki banyak variasi *blur* yang diberikan sehingga model bisa melakukan generalisasi dengan lebih baik dan dapat melakukan deteksi gambar dengan lebih baik pada skenario keadaan nyata.

Pengujian ketiga dilakukan untuk mendapatkan hasil pengaruh intensitas cahaya terhadap hasil deteksi yang dilakukan. Berdasarkan hasil pengujian tersebut didapat faktor kamera dan pemrosesan gambar oleh perangkat yang digunakan dalam melakukan deteksi sangat berpengaruh terhadap hasil yang diberikan. Hal tersebut terjadi karena banyak detail gambar yang hilang dan menyebabkan gambar sulit untuk dideteksi.

Pada pengujian ketiga didapatkan hasil pengaruh intensitas cahaya pada hasil deteksi yang menunjukkan semakin besar intensitas cahaya maka semakin baik juga akurasi yang didapatkan selama proses deteksi berlangsung. Pada saat troli diberikan lampu di sekitarnya untuk meningkatkan intensitas cahaya, terbukti bahwa hasil akurasi F1-Score meningkat sebesar 1.74 kali, mAP dengan *threshold* 0.5 meningkat sebesar 3.04 kali, dan mAP dengan *threshold* 0.5 meningkat sampai 0.95 sebesar 4.36 kali. Maka dari pengujian ketiga didapatkan kesimpulan bahwa penggunaan lampu di sekitar troli dapat memberikan pengaruh yang cukup besar pada hasil akurasi terutama pada kondisi pencahayaan yang gelap.

Pengujian keempat dilakukan untuk mendapatkan hasil pengaruh kecepatan produk terhadap hasil deteksi keseluruhan sistem yang dilakukan. Hasil pengujian menunjukkan bahwa kecepatan produk yang masuk atau keluar keranjang dengan

lebih pelan dapat terdeteksi dengan lebih baik juga. Hal tersebut dikarenakan *frame* dengan produk yang dapat ditangkap akan menjadi lebih banyak saat kecepatan produk menjadi lambat. Namun saat kecepatan produk menjadi lambat akan menjadi suatu kesulitan pada produk yang memiliki tinggi yang kurang karena tidak terdapat *motion blur* yang dapat membantu menambahkan tinggi produk. Lalu untuk produk yang memiliki ukuran yang kurang lebar juga akan sulit untuk dideteksi karena detail spesifik yang diberikan oleh produk akan lebih sedikit juga yang menyulitkan model saat melakukan deteksi. Dari pengujian ini juga bisa dikatakan kecepatan ideal yang dapat digunakan pada saat proses deteksi di komputer pengujian adalah hingga 36 cm/s dan untuk proses yang dilakukan di Raspberry Pi 4B adalah di bawah 7 cm/s. Hal tersebut didapat dari hasil *recognition rate* yang dihasilkan dari deteksi yang menunjukkan hubungan antara kecepatan dengan akurasi yang paling baik.

Pengujian kelima dilakukan untuk mengetahui pengaruh dalam melakukan penambahan *sampling rate* terhadap hasil deteksi produk. Hasil pengujian menunjukkan bahwa penambahan *sampling rate* dapat mempengaruhi hasil deteksi menjadi lebih baik karena *frame* yang terdapat produk menjadi lebih banyak. *Frame* yang terdapat produk menjadi lebih banyak dikarenakan proses pengambilan *frame* dilakukan di awal secara berurutan hingga mencapai jumlah *sampling rate* tanpa *delay* sehingga pergerakan produk yang tertangkap lebih baik. Dari pengujian ini juga bisa dikatakan kecepatan produk yang dapat digunakan pada saat proses deteksi di komputer pengujian dapat meningkat menjadi hingga 124 cm/s pada produk-produk dengan ukuran yang cukup lebar. Hal tersebut bisa terjadi juga dikarenakan *dataset* yang digunakan pada saat proses pelatihan model telah terdapat gambar yang memiliki *motion blur* sehingga saat produk bergerak cukup cepat masih dapat terdeteksi dengan cukup baik.

4.11. Analisis Dampak

Penelitian ini dilakukan untuk mendapatkan hasil yang nantinya dapat digunakan sebagai implementasi *smart trolley* di tempat perbelanjaan sebagai sistem *checkout* yang dapat mempermudah pengguna dalam berbelanja. Berdasarkan hasil pengujian yang menunjukkan model dapat melakukan deteksi dengan baik, maka

sistem sudah siap untuk diimplementasikan pada keadaan nyata di tempat perbelanjaan. Sistem dapat mempermudah pelanggan karena pada penelitian ini implementasi yang dibuat bisa menampilkan seluruh belanjaan yang ada di dalam troli secara otomatis sehingga sudah tidak diperlukan lagi adanya kasir. Namun sistem pada penelitian ini harus ditempatkan pada ruangan yang memiliki pencahayaan yang terang supaya akurasi model tidak menurun jauh dan produk dapat terdeteksi dengan baik.

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah dilakukan penelitian Rancang Bangun Metode *Object Detection* Yolov5 pada Raspberry Pi 4b untuk *Smart Trolley,* didapatkan beberapa kesimpulan:

1. Implementasi sistem *object detection* menggunakan model YOLOv5 pada Raspberry Pi 4B dapat diimplementasikan dengan baik untuk membangun sistem *smart trolley* yang dapat berjalan secara *real-time*.
2. Metode *object detection* menggunakan model YOLOv5 yang dibuat dapat mendeteksi produk yang keluar dan masuk keranjang belanja dengan baik terutama ketika dijalankan pada perangkat komputer pengujian.
3. Penggunaan *backbone swin transformer* tidak membuat model YOLOv5 memiliki akurasi yang lebih baik dari *backbone original* dengan nilai F1-Score model YOLOv5 *backbone original* sebesar 98.64% sedangkan YOLOv5 *backbone swin transformer* hanya sebesar 97.51%.
4. Penggunaan *backbone swin transformer* membuat model YOLOv5 dapat belajar lebih banyak fitur dibandingkan *backbone original* yang bisa mencegah terjadinya *overfitting*.
5. Lama pelatihan model YOLOv5 dengan *backbone swin transformer* lebih lama dibandingkan dengan *backbone original* dengan lama pelatihan YOLOv5 *backbone swin transformer* sebesar 8.679 menit per *epoch* sedangkan YOLOv5 *backbone original* hanya sebesar 4.762 menit per *epoch*.
6. Ukuran *file* TFLite yang dihasilkan oleh model YOLOv5 dengan *backbone swin transformer* lebih besar dibandingkan dengan *backbone original* dengan ukuran TFLite YOLOv5 *backbone swin transformer* sebesar 155.4 MB sedangkan YOLOv5 *backbone original* hanya sebesar 7.7 MB.
7. Model YOLOv5 dengan *backbone swin transformer* berjalan lebih lambat baik di komputer pengujian ataupun Raspberry Pi 4B dibandingkan dengan *backbone original* dengan YOLOv5 *backbone original* dapat berjalan 3.87

FPS di komputer pengujian dan 0.74 FPS di Raspberry Pi 4B sedangkan YOLOv5 *backbone swin transformer* hanya dapat berjalan 0.159 FPS di komputer pengujian dan 0.014 FPS di Raspbery Pi 4B.

8. Variasi kombinasi *dataset* bergerak dengan statis *blur* dapat menghasilkan model yang memiliki akurasi dan *loss* dengan rasio yang paling baik dengan nilai akurasi 99.53% pada fase pelatihan dan 99.44% pada fase testing.
9. Hasil F1-Score pada pengujian ketiga mengalami penurunan yang signifikan dikarenakan banyaknya kesalahan deteksi yang disebabkan oleh beberapa faktor yaitu resolusi kamera yang kurang baik, total *frame* yang ditangkap oleh kamera dalam waktu satu detik kurang dari 30, dan terdapat banyak *noise* yang mengganggu.
10. Penggunaan lampu untuk meningkatkan pencahayaan di sekitar wilayah deteksi di dalam troli dapat meningkatkan F1-Score hasil deteksi yang dilakukan hingga 63.55% terutama pada kondisi ruangan yang minim pencahayaan.
11. Variasi kecepatan produk saat masuk dan keluar keranjang menghasilkan kecepatan ideal yang dapat digunakan pada saat proses deteksi di komputer pengujian adalah hingga 36 cm/s dan untuk proses yang dilakukan di Raspberry Pi 4B adalah di bawah 7 cm/s.
12. Penambahan *sampling rate* membuat sistem dapat meningkatkan kecepatan produk di komputer pengujian dengan kecepatan hingga 124 cm/s pada produk-produk dengan ukuran yang cukup lebar.

5.2. Saran

Setelah dilakukan penelitian Rancang Bangun *Smart Trolley* Menggunakan Metode *Object Detection* Yolov5 pada Raspberry Pi 4b, didapatkan beberapa saran untuk pengembangan lebih lanjut yaitu:

1. Disarankan untuk menggunakan perangkat pemrosesan yang lebih baik untuk dapat mengimplementasikan model YOLOv5 dengan *backbone swin transformer*.

2. Disarankan untuk memperbanyak *dataset* untuk memperlihatkan potensi model YOLOv5 dengan *backbone swin transformer* yang saat ini masih berpotensi bisa menerima fitur yang lebih banyak.
3. Disarankan untuk memperbanyak variasi *dataset* seperti penambahan variasi sudut pandang, variasi jenis *background* gambar, menggabungkan beberapa objek pada satu gambar, dan memberikan variasi *blur* yang lebih beragam.
4. Disarankan untuk mengganti perangkat kamera yang digunakan dengan kamera yang memiliki resolusi lebih besar dan detail yang lebih baik.
5. Menggunakan lampu dengan intensitas cahaya yang lebih besar.
6. Melatih model dengan *epoch* yang lebih banyak dan menggunakan teknik penghentian *training* untuk mendapatkan hasil terbaik.
7. Menambahkan sensor jarak seperti *ultrasonic* supaya sistem pada pengujian kelima tidak perlu menekan tombol lagi setiap ingin mendeteksi.

DAFTAR REFERENSI

[1] V. Tyagi and A. K. Tyagi, "A Case Study on Consumer Buying Behavior towards Selected FMCG Products Effect of Organisational Culture on Employee Committment View project", doi: 10.13140/RG.2.2.16421.96485.

[2] H. Smith, "Supermarket Choice and Supermarket Competition in Market Equilibrium," 2004.

[3] D. Syarizka, "Belanja Online Marak, Toko Konvensional Tetap Punya Pelanggan. Ini Alasannya," *Bisnis.com*, Feb. 28, 2018. https://ekonomi.bisnis.com/read/20180228/100/744640/belanja-online-marak-toko-konvensional-tetap-punya-pelanggan.-ini-alasannya (accessed Dec. 23, 2021).

[4] R. B. Larson, "Supermarket self-checkout usage in the United States," *Services Marketing Quarterly*, 2019, doi: 10.1080/15332969.2019.1592861.

[5] S. Hedaux, "The cost of a queue: How much revenue are you losing to lines?," *Retail Customer Experience*, Apr. 22, 2020. https://www.retailcustomerexperience.com/blogs/the-cost-of-a-queue-how-much-revenue-are-you-losing-to-lines/ (accessed Dec. 23, 2021).

[6] M. Schögel and S. D. Lienhard, "Cashierless Stores – the New Way to the Customer?," *Marketing Review St. Gallen*, 2020.

[7] C. Bocanegra, M. A. (Amir) Khojastepour, M. Y. Arslan, E. Chai, S. Rangarajan, and K. R. Chowdhury, "RFGo: A Seamless Self-checkout System for Apparel Stores Using RFID," Sep. 2020, pp. 1–14. doi: 10.1145/3372224.3419211.

[8] S. R. Rupanagudi *et al.*, "A novel video processing based cost effective smart trolley system for supermarkets using FPGA," Feb. 2015. doi: 10.1109/ICCICT.2015.7045723.

[9] Y. Shinya, A. Ozeki, S. Mayumi, N. Motoyuki, I. Kenichi, and N. Narumitsu, "Special Issue on NEC Value Chain Innovation A Relaxed and Enjoyable Customer Experience, More Effi-cient Store Management-The Cashierless Future is Here."

[10] D. Ponte and S. Bonazzi, "Physical supermarkets and digital integration: acceptance of the cashierless concept," *Technology Analysis and Strategic Management*, 2021, doi: 10.1080/09537325.2021.1994942.

[11] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[12] B. Mahesh, "Machine Learning Algorithms-A Review Machine Learning Algorithms-A Review View project Self Flowing Generator View project Batta Mahesh Independent Researcher Machine Learning Algorithms-A Review," *International Journal of Science and Research*, 2018, doi: 10.21275/ART20203995.

[13] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," May 2019, [Online]. Available: http://arxiv.org/abs/1905.05055

[14] P. Ding, Y. Zhang, W. J. Deng, P. Jia, and A. Kuijper, "A light and faster regional convolutional neural network for object detection in optical remote sensing images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 141, pp. 208–218, Jul. 2018, doi: 10.1016/j.isprsjprs.2018.05.005.

[15] J. Yao, J. Qi, J. Zhang, H. Shao, J. Yang, and X. Li, "A real-time detection algorithm for kiwifruit defects based on yolov5," *Electronics (Switzerland)*, vol. 10, no. 14, Jul. 2021, doi: 10.3390/electronics10141711.

[16] Ultralytics, "YOLOv5," *Github*, Jun. 26, 2020. https://github.com/ultralytics/yolov5 (accessed Dec. 24, 2021).

[17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[18] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.14030

[19] G. S. Handelman *et al.*, "Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods," *American Journal of Roentgenology*, vol. 212, no. 1. American Roentgen Ray Society, pp. 38–43, Jan. 01, 2019. doi: 10.2214/AJR.18.20224.

[20] S. Yohanandan, "mAP (mean Average Precision) might confuse you!," *Towards Data Science*. https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 (accessed Jul. 09, 2022).

[21] B. Peng and L. Zhang, "LNCS 7574 - Evaluation of Image Segmentation Quality by Adaptive Ground Truth Composition," 2012.

[22] Roboflow, "Roboflow Features," *Roboflow*. https://roboflow.com/features (accessed Jul. 09, 2022).

[23] Roboflow, "YOLOv5 PyTorch TXT format," *Roboflow*. https://roboflow.com/formats/yolov5-pytorch-txt (accessed Dec. 25, 2021).

[24] Raspberry Pi, "What is a Raspberry Pi?" https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ (accessed Jul. 10, 2022).

[25] Raspberry Pi, "Raspberry Pi 4B," *Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-4-model-b (accessed Jul. 10, 2022).

[26] M. D. Mudaliar and N. Sivakumar, "IoT based real time energy monitoring system using Raspberry Pi," *Internet of Things (Netherlands)*, vol. 12, Dec. 2020, doi: 10.1016/j.iot.2020.100292.

[27] V. Weaver, "The GFLOPS/W of the various machines in the VMW Research Group." https://web.eece.maine.edu/~vweaver/group/green_machines.html (accessed Jul. 11, 2022).

[28] M. Ariyanto and P. D. Purnamasari, *2021 17th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering.* IEEE.

[29] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*, First Edition. O'Reilly Media, Inc., 2019.

[30] Y.-H. Cao and J. Wu, "A Random CNN Sees Objects: One Inductive Bias of CNN and Its Applications," Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.09259

[31] S. Cuenat and R. Couturier, "Convolutional Neural Network (CNN) vs Vision Transformer (ViT) for Digital Holography," Aug. 2021, [Online]. Available: http://arxiv.org/abs/2108.09147

[32] P. Radhakrishnan, "Why Transformers are Slowly Replacing CNNs in Computer Vision?" https://becominghuman.ai/transformers-in-vision-e2e87b739feb (accessed Jun. 20, 2022).