PERBANDINGAN SSD-MOBILENETV2 DENGAN SSD LITE-MOBILENETV2 MENGGUNAKAN RASPBERRY PI UNTUK KEAMANAN RUMAH SECARA *REAL-TIME*

SKRIPSI

Disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik (S.T)

Disusun oleh:

AAN MULANA

NPM.3332160067

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SULTAN AGENG TIRTAYASA

2023

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Dengan ini ditetapkan bahwa skripsi berikut:

Judul	: PERBANDINGAN SSD MOBILENETV2 DENGAN SSD LITE MOBILENETV2 MENGGUNAKAN RASPBERRY PI UNTUK KEAMANAN RUMAH SECARA REAL TIME
Nama Mahasiswa	: Aan Mulana
NIM	: 3332160067
Fakultas/Jurusan	: Teknik/Teknik Elektro

Menyatakan dengan sesungguhnya bahwa Skripsi di atas adalah benar-benar hasil karya asli saya dan tidak memuat hasil karya orang lain, kecuali dinyatakan melalui rujukan yang benar dan dapat dipertanggung jawabkan. Apabila dikemudian hari ditemukan hal-hal yang menunjukkan bahwa sebagian atau seluruh karya ini bukan karya saya, maka saya bersedia dituntut melalui hukum yang berlaku. Saya juga bersedia menanggung segala akibat hukum yang timbul dari pernyataan yang secara sadar dan sengaja saya nyatakan melalui lembar ini.

Cilegon, ...2023

NPM. 3332160067

LEMBAR PENGESAHAN

Dengan ini ditetapkan bahwa skripsi berikut:

Judul	:	PERBANDINGAN SSD MOBILENETV2 DENGAN SSD LITE MOBILENETV2 MENGGUNAKAN RASPBERRY PI UNTUK KEAMANAN RUMAH SECARA REAL TIME
Nama Mahasiswa	:	Aan Mulana
NIM	:	3332160067
Fakultas/Jurusan	:	Teknik/Teknik Elektro

Telah diuji dan dipertahankan pada tanggal 2023 melalui Sidang Skripsi di Fakultas Teknik Universitas Sultan Ageng Tritayasa Cilegon dinyatakan LULUS.

Dewan Penguji

		Tanda Tangan
Pembimbing I	: Dr. Irma Saraswati, S.SI., MT.	.....
Pembimbing II	: Fadil Muhammad, S.T., M.T.	.....
Penguji I	: Rian Fahrizal, ST., M.Eng.	.....
Penguji II	: Cakra Adipura W, S.T., M.T	.....

Mengetahui,
Ketua Jurusan Teknik Elektro

Dr. Romi Wiryadinata, M.Eng
NIP.19830703200912100

PRAKATA

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini, penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Sultan Ageng Tirtayasa. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Saya mengucapkan terima kasih kepada:

(1) Dr. Romi Wiryadinata, S.T., M.Eng., selaku Ketua Jurusan Teknik Elektro dan dosen pembimbing akademik yang telah memberikan bimbingan kepada saya selama masa perkuliahan,
(2) Dr. Irma Saraswati, S.SI., MT. selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(3) Fadil Muhammad, S.T., M.T. selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(4) Kedua orang tua tercinta serta seluruh keluarga yang telah memberikan nasehat, semangat, doa, dan materi yang tak terhingga nilainya.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Cilegon, 2023

Penulis

ABSTRAK

Aan Mulana

Teknik Elektro

Perbandingan SSD-MobilenetV2 dengan SSD Lite-MobilenetV2 menggunakan Raspberry Pi untuk Keamanan Rumah secara *Real-Time*

Penelitian ini dirancang untuk membuat sistem keamanan rumah dengan Raspberry Pi secara *real-time.* Sistem ini bekerja dengan mendeteksi manusia yang ditangkap oleh Pi camera kemudian diproses oleh Raspberry Pi. Pemprosesan oleh Raspberry Pi dengan bantuan dari *framework Tensorflow Lite* sehingga menghasilkan *output* orang yang dikenal dan tidak dikenal. Apabila mendeteksi orang tidak dikenal maka akan mengirimkan informasi ke Telegram berupa gambar dan teks. Mendeteksi manusia diperlukan training data. Training data menggunakan Google Colaboratory karena terdapat super GPU yang mempercepat proses training dan menghasilkan model SSD-MobilenetV2 dan SSD Lite-MobilenetV2. Pengujian model SSD-MobilenetV2 memiliki nilai rata-rata akurasi sebesar 97,35% dan fps 2,5. Pengujian model SSD Lite-MobilenetV2 memiliki nilai rata-rata akurasi sebesar 96,72% dan fps 5,6. Model SSD Lite-MobilenetV2 merupakan model pengujian yang memiki hasil data paling maksimal dalam pendeteksian manusia karena memiliki nilai fps 5,6 dan nilai akurasi tidak berbeda jauh dari SSD-MobilenetV2.

Kata kunci: Raspberry Pi, Google Colab, SSD-MobilenetV2, SSD Lite-MobilenetV2, Telegram, Tensorflow Lite.

ABSTRACT

Aan Mulana

Electrical Engineering

Comparison of SSD-MobilenetV2 with SSD Lite-MobilenetV2 using Raspberry PI for Home Security in Real-Time

This research is designed to create a real-time home security system using Raspberry Pi. The system operates by detecting humans captured by the Pi camera, then processed by the Raspberry Pi. Processing is carried out by the Raspberry Pi with the assistance of the Tensorflow Lite framework, resulting in outputs distinguishing between known and unknown person. If the unknown person is detected, the system will send the information to Telegram in the form of images and text. Detecting humans requires training data, which is accomplished using Google Colaboratory due to its super GPU that accelerates the training process and produces SSD-MobilenetV2 and SSD Lite-MobilenetV2 models. The result of SSD-MobilenetV2 model has an average accuracy value of 97.35% and a frame rate of 2.5 frames per second (fps). Meanwhile, the result of the SSD-Lite MobilenetV2 model has an average accuracy value of 96.72% and a frame rate of 5.6 fps. The optimal result for human detection model is SSD Lite-MobilenetV2, as it maintains a frame rate of 5.6 fps and its accuracy value is not significantly different from the SSD-MobilenetV2 model.

Kata kunci: Raspberry Pi, Google Colab, SSD-MobilenetV2, SSD Lite-MobilenetV2, Telegram, Tensorflow Lite.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi informasi dan komunikasi berkembang semakin pesat, salah satu perkembangan teknologi tersebut dimanfaatkan oleh beberapa pihak untuk memanfaatkan potensi yang ada dengan menggunakan inovasi, yang salah satunya adalah teknologi informasi dalam bidang keamanan lingkungan contohnya yaitu keamanan rumah.

Keamanan rumah diperlukan karena banyak terjadi kasus pencurian terlebih pada malam hari. Beberapa lingkungan perumahan sudah memiliki petugas keamanan, namun keterbatasan manusia dapat menjadi celah bagi pelaku pencurian. Jumlah insiden kejahatan terhadap hak milik tanpa kekerasan selama lima tahun terakhir cenderung mengalami penurunan. Berdasarkan data tahun 2019, terdapat 78.330 insiden, yang kemudian berkurang menjadi 73.264 insiden pada tahun 2020, dan terus menurun menjadi 69.347 insiden pada tahun 2021. Tindak pidana pencurian tanpa penggunaan kekerasan merupakan jenis kejahatan dengan jumlah paling banyak setiap tahunnya [1].

Penelitian ini bertempat di pekarangan rumah. Lokasi ini dipilih karena terdapat aktifitas lalu-lalang manusia. Menentukan pendeteksian manusia yang akan memasuki rumah maka dilakukan pendeteksian khusus untuk orang rumah dan melakukan pendeteksian selain orang rumah. Pendeteksian nantinya akan difokuskan kepada selain orang rumah atau orang asing. Penelitian ini merancang sistem pendeteksian manusia yang meliputi orang rumah atau bukan. Jika bukan orang dikenal maka mengirimkan informasi ke Telegram berupa gambar dan teks secara *real-time*. Sistem yang dibuat pada penelitian ini adalah dengan memanfaatkan metode pengolahan citra.

Salah satu metode yang dapat diterapkan untuk mengolah informasi yang terdapat dalam gambar atau video adalah melalui pengolahan citra [2]. Proses pengolahan citra melibatkan penggunaan kamera untuk merekam dan memonitor aktivitas pengendara, sedangkan data video yang dihasilkan oleh kamera tersebut

dianalisis dan diolah melalui komputer [2]. Jenis kamera yang digunakan bisa berupa kamera CCTV atau modul kamera lainnya yang memiliki kualitas video yang tinggi. Pengaturan sudut pengambilan gambar juga perlu diperhatikan agar hasil gambar yang diperoleh optimal.

Berbagai penelitian menawarkan sistem yang mampu menambah tingkat keamanan rumah untuk pendeteksian manusia. Penelitian sebelumnya menggunakan menggunakan metode pengolahan citra dengan *framework* Tensorflow lite dan model SSD-MobilenetV2 [2], sistem RO [3], metode *waterfall* [4], sistem PIR [5], metode LBP [6] dan sistem *Internet of Things (Iot)* [7]. Tujuan dari penelitian ini bukan hanya untuk mendeteksi manusia, tetapi juga membandingkan model yang terbaik untuk digunakan dalam keamanan rumah secara *real-time*. Penelitian ini juga melakukan pengembangan dari penelitian sebelumnya yaitu mengirimkan hasil deteksi manusia ke telegram berupa gambar dan teks secara *real-time*.

Penelitian ini menggunakan model SSD-MobilenetV2 dan model SSD Lite-MobilenetV2. SSD-MobilenetV2 merupakan salah satu arsitektur CNN yang berguna untuk mendeteksi beberapa objek serta mengekstrasi secara efisen dan cepat. Arsitektur CNN (*Convolutional Neural Network*) untuk mengatasi kebutuhan komputasi berlebih [8]. SSD Lite-MobilenetV2 masih dengan arsitektur yang sama namun memiliki keunggulan dapat mempercepat proses komputasi secara keseluruhan dan mengekstrasi fitur [9]. Sistem dalam penelitian ini memiliki program dan hardware untuk melakukan proses klasifikasi. Program yang digunakan menggunakan bahasa Python dengan bantuan *library* OpenCV, Tensorflow Lite, Numpy, Math, Time, dan Telepot [10]. *Hardware* yang digunakan adalah Raspberry Pi dan Pi *camera*. Raspberry pi merupakan sebuah komputer berukuran mini sebesar kartu kredit yang dikembangkan di Inggris oleh Raspberry *Foundation* [11]. Sistem ini dapat melakukan deteksi manusia dengan mengambil gambar menggunakan Pi camera kemudian hasil gambar tersebut diproses secara *real-time* pada Raspberry Pi dengan menggunakan program yang telah dibuat, sehingga menghasilkan gambar manusia. Sistem *real-time* adalah bentuk khusus dari sistem *online*. Sistem dianggap *real-time* jika tidak hanya mengutamakan ketepatan dalam proses, tetapi juga interval waktu proses tersebut dilakukan [12].

Modul Pi *Camera* merupakan salah satu aksesoris pendukung untuk Raspberry Pi dan memiliki kemampuan untuk mengambil gambar dengan resolusi 1080p dan 720p, serta merekam video dengan ukuran 640x480p [13], setelah mendapatkan gambar dari manusia tersebut kemudian mengirimkan informasi tersebut ke Telegram [14]. Proses deteksi manusia mengunakan *framework* dari tensorflow lite dan membandingkan model SSD-MobilenetV2 dengan SSD Lite-MobilenetV2. Kedua model tersebut harus di training terlebih dahulu agar dapat melakukan pendeteksian manusia dan dapat mengenali manusia. Sebelum melakukan proses *training* diperlukan proses *preprocessing training* seperti melakukan pengambilan gambar kendaraan dari video arus lalu lintas, normalisasi gambar, labeling gambar, konversi XML ke CSV, membuat *file record* dan membuat *label map*. Setelah melakukan proses *preprocessing training* tahap berikutnya adalah melakukan proses *training*. Proses *training* dilakukan di Google Colaboratory karna disana terdapat super GPU sehingga mempercepat proses *training* [15]. Proses *training* menghasilkan model SSD-MobilenetV2 dan SSD Lite-MobilenetV2 kemudian dilakukan optimalisasi untuk mengecilkan ukuran *file* model tersebut sehingga mempercepat proses deteksi manusia [16]. Penelitian ini dilakukan perbandingan model SSD MobilenetV2 dan model SSD Lite-MobilenetV2 untuk menghasilkan model yang terbaik untuk sistem keamanan rumah secara *real-time*.

1.2 Rumusan Masalah

Rumusan permasalahan dalam penelitian ini yaitu:

1. Bagaimana proses kerja pendeteksian manusia menggunakan Raspberry Pi secara *real-time*?
2. Bagaimana proses pengiriman data hasil deteksi ke telegram?
3. Bagaimana tingkat akurasi deteksi manusia menggunakan model SSD-MobilenetV2 dan SSD Lite-MobilenetV2 pada Raspberry Pi?

1.3 Tujuan Penelitian

Tujuan penelitian ini yaitu:

1. Menganalisa proses kerja deteksi manusia Raspberry Pi secara *real-time*.
2. Menganalisa proses pengiriman data hasil deteksi manusia ke Telegram.

3. Membandingkan dan menganalisa tingkat akurasi deteksi manusia menggunakan model SSD-MobilenetV2 dan SSD Lite-MobilenetV2 pada Raspberry Pi.

1.4 Manfaat Penelitian

Merujuk pada perumusan masalah dan tujuan penelitian, manfaat dari penelitian ini yaitu untuk membuat program keamanan rumah secara *real-time* berbasis vision menggunakan Raspberry pi dan mengetahui tingkat akurasi dan kecepatan dalam pendeteksian manusia menggunakan model SSD-MobilenetV2 dan SSD Lite-MobilenetV2 pada Raspberry Pi.

1.5 Batasan Penelitian

Batasan masalah pada penelitian ini adalah sebagai berikut :

1. *Database* yang digunakan untuk pengiriman data adalah Telegram.
2. Menggunakan *Pi camera* untuk pengambilan data latih dan pengujian.
3. Menggunakan modem *Wi-Fi* sebagai koneksi internetnya.
4. Menggunakan *framework* Tensorflow Lite.
5. Intensitas cahaya tempat penelitian adalah 50 lux.
6. Lokasi pengambilan data latih dan pengujian berlokasi di depan rumah.

1.6 Sistematika Penulisan

Penulisan dibagi menjadi beberapa pokok permasalahan untuk memperjelas pembahasan penelitian yang secara garis besar sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi mengenai latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi mengenai tinjauan pustaka dan dasar teori, didalamnya memuat kajian-kajian dari hasil artikel publikasi terkait dengan topik penelitian, dan landasan - landasan teori yang menunjang penyelesaian penelitian diantara lain Sistem *real-time*, python, OpenCV, Raspberry Pi, Pi camera, Modem Wi-fi, Deteksi Objek, Google Colaboratory dan Telegram.

BAB III METODOLOGI PENELITIAN

Bab ini berisi mengenai proses Metode penelitian, Instrumen penelitian, cara kerja alat, perangkat dan spesifikasi alat yang digunakan dalam pembuatan alat, baik perangkat keras (*hardware*) atau perangkat lunak (*software*), serta tempat dan waktu penelitian.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi mengenai hasil dari penelitian serta analisa terhadap hasil penelitian yang dikaitkan dengan tinjauan pustaka dan metodologi penelitian.

BAB V PENUTUP

Bab ini berisi kesimpulan dari penelitian dan saran untuk pengembangan penelitian lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Sistem *Real-Time*

Suatu sistem yang dapat mengendalikan sistem fisik dikenal sebagai sistem *real-time* [11]. Sistem *real-time* adalah bentuk khusus dari sistem *online*. Sistem dianggap *real-time* jika tidak hanya mengutamakan ketepatan dalam proses, tetapi juga interval waktu proses tersebut dilakukan. Sistem *real-time* adalah sistem yang menggunakan *deadline*, yaitu pekerjaan harus selesai padajangka waktu tertentu. Sistem *real-time*, digunakan batasan waktu. Sistem dinyatakan gagal jika melewati batasan yang ada. Sistem *real-time* telah banyak digunakan dalam berbagai macam aplikasi. Sistem tersebut ditanam didalam alat khusus seperti di kamera, mp3 *players*, serta di pesawat dajend [12].

Terdapat dua bentuk sistem *real-time*, yaitu sistem *hard real-time* dan sistem *soft real-time*. Sistem *hard real-time* bertujuan untuk menjamin tugas kritis diselesaikan tepat waktu. Sistem ini penyimpan sekunder terbatas atau tidak digunakan, data langsung dikirim ke *memory* atau *read-only memory* (ROM) dalam waktu singkat.

Sistem *hard real-time* terjadi konflik pada sistem *time sharing* dan tidak didukung oleh sistem operasi tujuan umum. Bentuk lainnya adalah *soft real-time* dimana tugas kritis mendapat prioritas lebih tinggi dari tugas lain dan setelah satu *task* selesai maka *task* berprioritas ini diselesaikan. Sistem ini terbatas pada industri pengontrol robot yang sangat berguna pada aplikasi multimedia dan *virtual reality* yang membutuhkan fitur sistem operasi tertentu [12].

Suatu sistem dikatakan *real-time* jika sistem tersebut dapat mendukung eksekusi program atau aplikasi dengan waktu yang memiliki batasan. Kelebihan suatu sistem *real-time*, yaitu :

a. Memenuhi *deadline* dan memiliki batasan waktu .
b. Memprediksi waktu yang terjadi.
c. Terfokus pada hal-hal yang penting saja, yang tidak penting tidak dikerjakan sehingga dapat menemukan tingkat efisiensi waktu.Selain kelebihan, ada

beberapa kekurangan sistem *real-time*, yaitu :

a. Hanya memiliki satu tujuan, seperti mentransfer sebuah lagu dari komputer ke *music player*.
b. Kebanyakan sistem memiliki *physical space* yang terbatas.
c. Sistem harus memenuhi persyaratan waktu yang telah ditentukan dengan menggunakan algoritma.

2.2 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python juga didukung oleh komunitas yang besar [9].

Python mendukung multi paradigma pemrograman, utamanya namun tidak dibatasi pada pemrograman berorientasi objek, pemrograman imperatif, dan pemograman fungsional. Salah satu fitur yang tersedia pada Python adalah sebagai bahasa pemrograman dinamis yang dilengkapi manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, Python umumnya digunakan sebagai bahasa *script* meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa *script*. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai *platform* sistem operasi [9].

Python memiliki proses yang lebih lambat jika dibandingkan dengan bahasa pemrograman lainnnya, seperti C/C++. Python memiliki fitur penting, yaitu dapat dengan mudah melakukan *extend* dengan bahasa C/C++, jadi dengan demikian dapat menuliskan kode program yang membutuhkan komputasi tinggi dalam bahasa C/C++ dan kemudian dibuat Python *wrapper*, sehingga kode C/C+ tersebut dapat digunakan sebagai modul dalam bahasa Python. Terdapat banyak sekali referensi bahasa Python dibandingkan dengan bahasa pemrograman yang lain dan sangat cocok digunakan untuk pemula yang ingin belajar *coding*. Bahasa Python sangat cocok digunakan untuk melakukan deteksi objek, data *mining*, klasifikasi, dan lain-lain [9].

2.3 OpenCV

OpenCV (*Open Source Computer Vision Library*), adalah sebuah *library open source* yang dikembangkan oleh Intel yang memiliki fokus untuk menyederhanakan programing terkait citra digital. OpenCV sudah mempunyai banyak fitur, antara lain : pengenalan wajah, pelacakan wajah, deteksi wajah, Kalman *filtering*, dan berbagai jenis metode AI (*Artificial Intellegence*) dan menyediakan berbagai algoritma sederhana terkait *Computer Vision* untuk *low level* API [10].

OpenCV merupakan *open source computer vision library* untuk bahasa pemrograman C/C++, dan telah dikembangkan ke Python, Java, Matlab. Intel meluncurkan versi pertama dari OpenCV pada 1999, dan awalnya memerlukan *library* dari Intel *Image Processing Library*. *Dependecy* tersebut ahkirnya dihilangkan sehingga terciptalah OpenCV seperti yang sekarang sebagai *standalone library*. OpenCV mendukung *multiplatform*, dapat mendukung baik windows atau linux, dan sekarang telah mendukung MacOSX dan Android [10]. OpenCV memiliki fitur-fitur pada *library* OpenCV antara lain:

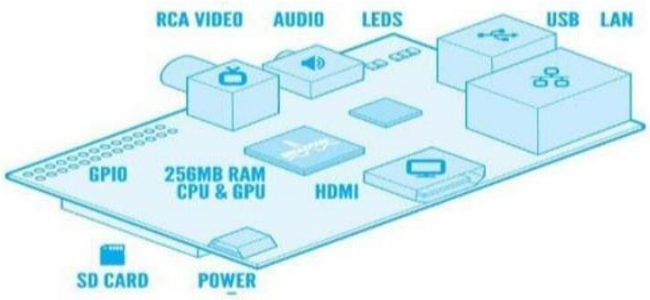
1. Manipulasi data gambar (alokasi memori, melepaskan memori, *copy* gambar, *setting* serta konversi gambar).
2. Metode untuk AI dan *machine learning*.
3. Memproses gambar/video.
4. Sampling gambar dan transformasi.
5. Kalibrasi kamera.
6. Pendeteksian gerak.
7. Manipulasi matriks dan vektor serta terdapat juga *routines linear algebra* (*products*, *solvers*, *eigenvalues*, SVD).
8. *Image processing* dasar (*filtering*, *edge detection*, pendeteksian tepi, sampling dan interpolasi, konversi warna, operasi morfologi, histograms, *image pyramids*).
9. Analisis struktural.
10. Pengenalan objek.
11. Basic GUI (display gambar/video, mouse/keyboard kontrol, *scrollbar*).
12. *Image Labelling* (*line*, *conic*, *polygon*, *text drawing*).

2.4 Raspberry Pi

Raspberry pi merupakan sebuah komputer berukuran mini sebesar kartu kredit yang dikembangkan di Inggris oleh Raspberry *Foundation*. Gagasan dibalik sebuah computer kecil dan murah untuk anak-anak muncul pada tahun 2006. Perkembangannya Raspberry Pi mengalami 7 tahap perubahan model sejalan dengan tahap penelitian hingga tahap produksi yang sampai pada masarakat pengguna. Berikut ini adalah tahap perubahan model pada Raspberry Pi [11]:

1. Raspberry Pi edisi 2006
2. Raspberry Pi USB Prototype Board
3. Raspberry Pi Alpha Board
4. Raspberry Pi Beta Production Board
5. Raspberry Pi 1st Production Board
6. Raspberry Pi Model-B Full Production Board
7. Raspberry Pi Model-A Full Production Board

Komponen dalam Raspberry Pi model B+ yang digunakan pada penelitian ini dapat dilihat pada Gambar 2.1



Gambar 2.1 Komponen Raspberry Pi Model B+

1. CPU

   Ditenagai oleh *processor* ARM11 dengan *default clock* sebesar 700 MHz, Raspberry Pi cukup mumpuni untuk menjalakan tugasnya sebagai Komputer berukuran relatif kecil, baik dijadikan portabel atau *desktop*. Pihak Raspberry Pi mengizinkan penggunaan mode turbo, yaitu *overclock* hingga 2GHz.

2. GPU

Sudah mengusung teknologi Open GL ES 2.0, mendukung resolusi video hingga 1080p dengan slot HDMI, H.264 *high-profile encode*/*decode*. Kapasitas grafis pada Raspberry Pi setara dengan Xbox 1.

3. *Memory*

Kapasitas memori 512 MB cukup bertenaga pada mode *console* atau windows. Sistem memori pada Raspberry Pi bersifat *shared* dengan GPU, sehingga diperlukan pembagian kapasitas memori untuk *graphic chip*

4. Port USB

Raspberry Pi Model B+ memiliki 2 port USB sehingga mendukung kompatibilitas dengan perangkat universal lain berbasis USB. Juga terdapat USB *powered hub* tambahan yang sudah dilengkapi daya eksternal dengan rata – rata arus maksimal sebesar 2A.

5. *Micro USB Power*

Bagian ini menjadi sumber utama bagi Raspberry Pi untuk mendapatkan sumber daya. Namun dapat juga menggunakan USB *powered hub* yang dihubungkan melalui port USB tanpa harus menghubungkan power lagi. Sumber daya yang direkomendasikan yaitu sebesar 5V dan minimal arus 700 mA.

6. SD Card

Raspberry Pi menggunakan SD Card sebagai media penimpanan utama. Disarankan menggunakan SD Card berkapasitas 4 GB kelas 4.

7. Audio Jack

Berfungsi sebagai keluaran suara.

8. *Ethernet*

Berfungsi menghubungkan Raspberry Pi ke dalam jaringan internet. Juga dapat digunakan untuk mengakses Raspberry Pi secara *remote*.

9. *General Purpose Input Output* (GPIO)

Raspberry Pi Model B+ mempunyai *pin* sebanyak 40 buah. *Pin* ini digunakan untuk proyek yang berhubungan dengan *hardware*.

2.5 *Pi camera*

Modul Pi *Camera* merupakan salah satu aksesoris pendukung untuk Raspberry Pi yang dirancang dan diproduksi oleh Raspberry Pi *Foundation* di UK.

Kamera ini memiliki resolusi 5 MP dan dilengkapi dengan kabel data fleksibel yang berfungsi untuk menghubungkannya pada konektor CSI yang terletak di antara *port ethernet* dan *port* HDMI pada papan Raspberry Pi. Pi *Camera* memiliki kemampuan untuk mengambil gambar dengan resolusi 1080p dan 720p, serta merekam video dengan ukuran 640x480p. Modul kamera ini memiliki dimensi 25mm x 20mm x 9mm. Kamera Pi dapat berfungsi dengan semua model Raspberry Pi [13].

2.6 Modem Wi-Fi

Wi-Fi adalah singkatan dari *Wireless Fidelity* yang artinya jaringan nirkabel yang dapat memancarkan koneksi internet. Modem merupakan perangkat keras singkatan dari *Modulator Demodulator* yang memiliki fungsi untuk mengubah 2 sinyal berbeda yaitu dari sinyal digital menjadi sinyal analog. Kedua istilah tersebut digabung menjadi MiFi yang artinya sebuah perangkat keras yang memancarkan sambungan internet tanpa menggunakan kabel.

Mifi dapat terhubung ke operator seluler dan menyediakan akses internet untuk perangkat lainnya yang terhubung. Salah satu kelemahan MiFi adalah terbatasnya jumlah komputer *client* yang bisa terhubung dan mengakses internet. MiFi dapat dengan mudah membuat *hotspot* Wi-Fi sendiri yang terhubung ke internet melalui jaringan seluler [17].

2.7 Deteksi Objek

Disaat manusia melirik pada sebuah gambar, otak manusia langsung dapat mengenali objek yang berada pada gambar, letak objek tersebut, dan kondisi interaksi yang terjadi. Sistem visual manusia cepat dan akurat, memungkin untuk melakukan tugas-tugas kompleks. Algoritma deteksi objek yang cepat dan akurat memungkinkan komputer dapat melakukan hal serupa hingga berpotensi untuk menyelesaikan tugas secara umum. Deteksi objek dalam digital *image processing* adalah suatu proses yang digunakan untuk menetukan keberadaan objek tertentu di dalam suatu citra digital.

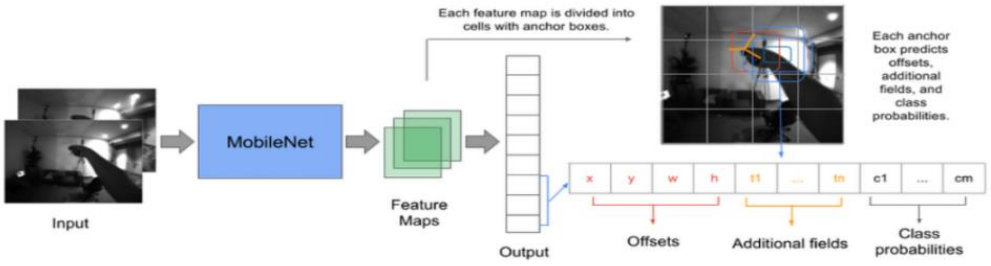
Proses deteksi tersebut dapat dilakukan dengan berbagai macam metode

yang umumnya melakukan pembacaan fitur-fitur dari seluruh objek pada citra *input*. Fitur dari objek pada citra *input* tersebut dibandingkan dengan fitur darimodel yang digunakan atau *template*. Hasil perbandingan tersebut dapat digunakan untuk menentukan apakah suatu objek terdeteksi sebagai *template* yang dimaksud atau tidak.

Sistem deteksi objek perlu dilatih dan diuji *dataset* dengan *bounding box* dan diberi label untuk kelas disetiap objek untuk proses pengenalan. Mencapai bagian ni, ada banyak *dataset* untuk menghasilkan model *Deep Learning* seperti Pascal-VOC. Penelitian ini pendeteksian manusia menggunakan metode *cnn* dengan model SSD-MobilenetV2 dan *framework* Tensorflow Lite [14].

2.7.1 MobileNet

MobileNet merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang dapat digunakan untuk mengatasi kebutuhan sumber komputasi berlebih. Google membangun MobileNet atas kebutuhan arsitektur CNN yang dapat digunakan untuk ponsel atau sistem tertanam. Perbedaan mendasar antara arsitektur MobileNet dan arsitektur CNN pada umumnya adalah penggunaan lapisan atau layer konvolusi dengan ketebalan filter yang sesuai dengan ketebalan dari *input image*. MobileNet membagi konvolusi menjadi *depthwise convolution* dan *pointwise convolution*.

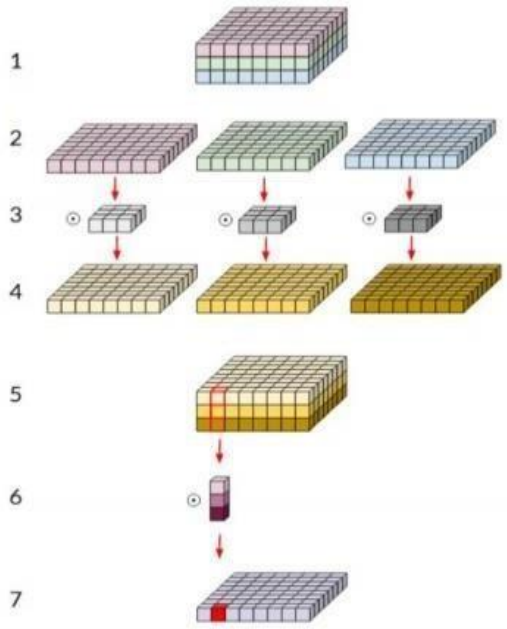


Gambar 2.2 Alur Kerja Mobilenet

Arsitektur MobileNet memanfaatkan *Batch Normalization* (BN) dan *Rectified-Linear unit* (ReLU) untuk *depthwise convolution* dan *pointwise convolution*. MobileNet dibangun di atas arsitektur jaringan yang efisien dengan menggunakan konvolusi yang dapat dipisahkan secara mendalam untuk menghasilkan *Deep Neural Network* yang ringan [8]. Alur kerja MobileNet dapat

dilihat pada Gambar 2.2.

DSC menggantikan konvolusi standard dengan 2 tahap operasi. Pertama adalah *depthwise convolution* dimana setiap filter $DF \times DF$ hanya melakukan proses *filter* terhadap sebuah *feature map input* secara mendalam. Kedua adalah *pointwise convolution* yang merupakan 1×1 *convolution layer* yang digunakan untuk menggabungkan jalur informasi dari *depthwise layer* [6]. DSC menyebabkan jalur konvolusi menjadi jauh lebih efesien dengan menggunakan pada Gambar 2.3.



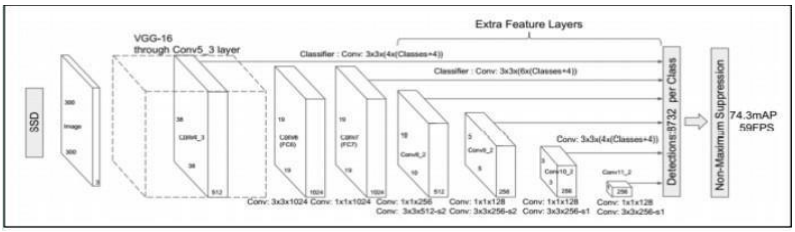
Gambar 2.3 Konvolusi *Depthwise* dan *Pointwise*

2.7.2 *Single Shot MultiBox Detector* (SSD)

Single Shot Detector (SSD) adalah sebuah metode untuk mengenali atau mendeteksi sebuah objek pada suatu gambar dengan menggunakan *single deep neural network* dan salah satu algoritma deteksi objek yang paling populer karena kemudahan implementasi, akurasi yang baik untuk rasio yang dibutuhkan komputasi. SSD hanya perlu mengambil satu bidikan tunggal untuk mendeteksi beberapa objek didalam gambar.

SSD ini termasuk kedalam deteksi objek secara *real-time* lebih intuitif daripada rekan-rekannya seperti R-CNN, *Fast* R-CNN, Faster R-CNN dan *You Only Look Once* (YOLO). Menjadi sederhana dalam desain, implementasinya lebih langsung dari GPU dan sudut pandang kerangka kerja pembelajaran yang dalam

dan dengan demikian melakukan pengangkatan berat deteksi dengan kecepatan kilat. Terdapat gambar arsitektur *single shot detector* pada Gambar 2.4.



Gambar 2.4 Arsitektur *Single Shot Detector*

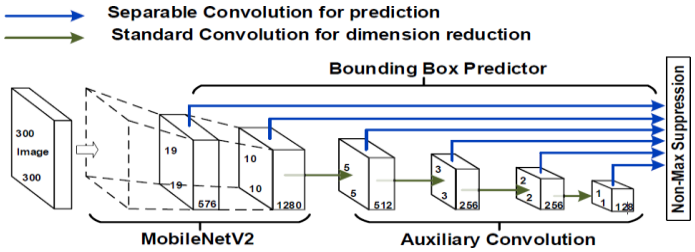
Arstitektur SSD termasuk jenis *Convolutation Neural Network. Convolutional Neural Network* (CNN) adalah salah satu jenis *Neural Network* yang biasa digunakan pada data *image*. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah gambar. Arsitektur dari CNN dibagimenjadi 2 bagian besar, *Feature Extraction Layer* dan *Convolutional Layer*, dimana pada bagian *Feature Extraction Layer* ini adalah melakukan *encoding* dari sebuah *image* menjadi *features* yang berupa angka-angka yang merepresentasikan gambar tersebut, sedangkan dibagian *Convolutional Layer* terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah *filter* dengan panjang dan tinggi (*pixels*). Secara matematis, *Convolutional layer* atau dalam Bahasa Indonesianya konvolusi adalah integral yang mencerminkan jumlah lingkaran dari sebuah sudut fungsi F yang digeser atas fungsi G sehingga menghasilkan fungsi H [18].

2.7.3 SSD Lite-MobilenetV2

SSD Lite adalah varian SSD yang cocok untuk perangkat *mobile*, dimana reguler konvolusi dalam kotak pendeteksi diganti dengan *depth-wise* konvolusion. Berdasarkan *framework* SSD Lite, SSD Lite-MobilenetV2 yang diusulkan untuk menggunakan MobilenetV2 sebagai jaringan dasar menggambarkan struktur jaringan SSD Lite-MobilenetV2 [9]. Terdapat rangkaian gambar *Architecture of SSD Lite* pada Gambar 2.4.

Detektor satu tahap yang dikenal sebagai *single shot detectors*, seperti SqueezeDet, SSD, YOLO, hanya memerlukan satu langkah melalui jaringan untuk memprediksi semua kotak pembatas, membuatnya cocok untuk penggunaan inferensi yang efisien pada perangkat terbatas seperti perangkat tepi. Kategori ini,

SSDLite merupakan varian SSD yang efisien dan SSDLite adalah model umum yang digunakan untuk berbagai aplikasi, termasuk deteksi plat nomor, mobil, wajah, dan tangan.



Gambar 2.5 *Architecture Of SSD Lite* [9].

SSDLite adalah model berkegunaan umum; beberapa aplikasi dari model ini meliputi deteksi plat nomor, deteksi mobil, deteksi wajah, deteksi tangan, hanya beberapa di antaranya.

2.8 Google Colaboratory

Google Colaboratory adalah sebuah proyek yang bertujuan untuk menyebarkan pendidikan dan penelitian *machine learning* [17]. Buku catatan kolaboratoris didasarkan pada Jupyter dan berfungsi sebagai objek Google Dokumen dapat dibagikan dan pengguna dapat berkolaborasi di buku catatan yang sama. Colaboratory menyediakan *runtime* Python 2 dan 3 yang telah dikonfigurasi sebelumnya dengan pustaka *machine learning* dan *artificial intelligence* yang penting, seperti TensorFlow dan Matplotlib. Mesin virtual di bawah *runtime* (VM) dinonaktifkan setelah jangka waktu tertentu, dan semua data dan konfigurasi pengguna hilang, namun *notebook* tersebut dipertahankan dan memungkinkan untuk mentransfer *file* dari *hard disk* VM ke akun Google Drive pengguna [15].

Google Colaboratory mempunyai 4 virtual GPU yaitu Nvidia Tesla K80s, T4s, P4s, dan P100s. Hanya GPU Nvidia Tesla K80s yang dapat digunakan secara gratis, sisanya harus menggunakan Google Colaboratory yang versi berbayar. Penggunaan Google Colaboratory yang gratis disediakan juga *memory* GPU sebesar 12 GB, RAM sebesar 25 GB, dan *termporary disk* sebesar 107 GB. Penggunaan Google Colaboratory yang versi gratis maka waktu proses yang dapat digunakan hanya selama 12 jam, setelah itu *cooldown* selama 12 jam. Masa *cooldown* tidak bisa menggunakan Google Colaboratory sehingga harus menunggu

untuk menggunakannya kembali.

2.9 Optimalisasi Model

Perangkat dengan sistem tertanam seperti *handphone* dan Raspberry Pi sering kali memiliki memori atau daya komputasi yang terbatas. Berbagai pengoptimalan dapat diterapkan ke model agar dapat dijalankan dalam batasan ini. Beberapa pengoptimalan memungkinkan penggunaan perangkat keras khusus untuk inferensi yang dipercepat [16].

Tujuan dari optimalisasi model adalah untuk mengurangi ukuran *file*, mengurangi latensi, dan kompatibilitas akselerator. Optimalisasi model bekerja dengan mengurangi ketepatan dari angka yang digunakan untuk merepresentasikan parameter model. Efek yang terjadi adalah model ukuran menjadi lebih kecil dan komputasi menjadi lebih cepat.

2.10 Telegram

Telegram adalah aplikasi pesan instan berbasis *cloud* yang fokus pada kecepatan dan keamanan. Telegram dirancang untuk memudahkan pengguna saling berkirim pesan teks, audio, video, gambar dan *sticker* dengan aman [14].

Secara *default*, seluruh konten yang ditransfer dienskripsi berstandar internasional. Pesan yang terkirim sepenuhnya aman dari pihak ketiga bahkan dari Telegram. Bukan hanya teks, gambar dan video, Telegram juga bisa jadi sarana untuk mengirimkan dokumen, musik, berkas zip, lokasi *real-time* dan kontak yang tersimpan keperangkat orang lain. Telegram merupakan aplikasi berbasis *cloud*, yang memudahkan penggunanya dapat mengakses satu *account* Telegram dari perangkat yang berbeda dan secara bersamaan [22].

2.11 Evaluasi dan Validasi Algoritma Klasifikasi Data *Mining*

Pengukuran akurasi algoritma klasifikasi dapat menggunakan metode yang *cross validation*, *confusion matrix*, dan kurva ROC (*Receiver Operating Characteristic*). Mengembangkan aplikasi (*development*) berdasarkan model yang dibuat, digunakan *Rapid Miner.*

1. *Cross Validation*

Cross validation adalah pengujian standar yang dilakukan untuk memprediksi *error rate*. Data *training* yang ada dibagi secara acak ke dalam beberapa bagian dengan perbandingan yang sama, kemudian *error rate* dihitung bagian demi bagian, selanjutnya menghitung rata-rata seluruh *error rate* untuk mendapatkan *error rate* secara keseluruhan.

2. *Confusion Matrix*

Metode ini hanya menggunakan Tabel matriks seperti pada Tabel 2.1, jika *dataset* hanya terdiri dari dua kelas, kelas yang satu dianggap sebagai positif dan yang lainnya negatif [23]. Evaluasi dengan *confusion matrix* menghasilkan nilai *accuracy,* F-*Score, precison, dan recall*. *Accuracy* merupakan prediksi benar dengan keseluruhan data, sedangkan *precision* adalah rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. *Recall* adalah rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. F-*Score* merupakan perbandingan rata-rata presisi dan *recall* yang dibobotkan. Menghitung beberapa parameter didalam *confusion matrix* menggunakan persamaan 2.1 sampai dengan Persamaan 2.4

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.1}$$

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.2}$$

$$F - Score = \frac{2 \times (\mathrm{Precision} \times \mathrm{Recall})}{\mathrm{Precision} + \mathrm{Recall}} \tag{2.3}$$

$$\text{Akurasi} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP} + \mathrm{TN}} \times 100\% \tag{2.4}$$

Persamaan 2.1 sampai 2.4 terdapat keterangan TP, FP, FN dan TN. TP menyatakan *true positives,* FP menyatakan *false positives,* FN menyatakan *false negatives* dan TN menyatakan *true negatives*. Model *confusion matrix* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Model *Confusion Matrix*

Correct Classification	*Classifie as*	
	+	-
+	*True Positives*	*False Negatives*
-	*False Positives*	*True Negatives*

3. Kurva ROC

Kurva ROC menunjukkan akurasi dan membandingkan klasifikasi secara visual. ROC mengekspresikan *confusion matrix*. ROC adalah grafik dua dimensi dengan *false positives* sebagai garis *horizontal* dan *true positives* sebagai garis *vertical* (Vecellis, 2009). *The area under curve* (AUC) dihitung untuk mengukur perbedaan performansi metode yang digunakan. AUC dihitung menggunakan persamaan 2.5 dan 2.6.

$$\theta^r = \frac{1}{mn}\sum_{j=1}^{n}\sum_{i=1}^{m}\psi(xi^r, xj^r) \tag{2.5}$$

dimana:

$$(X,Y) = \begin{cases} 1 & Y < X \\ \frac{1}{2} & Y = X \\ 0 & Y > X \end{cases} \tag{2.6}$$

K merupakan jumlah algoritma klasifikasi yang dikomparasi, X merupakan *output* positif, dan Y merupakan *output* negatif.

2.12 Kajian Pustaka

Bagian pendahuluan kajian pustaka ini membahas tinjauan literatur sebelumnya yang berhubungan dengan penelitian tentang sistem deteksi manusia untuk meningkatkan keamanan rumah. Sistem keamanan rumah menggunakan raspberry pi sebagai perangkat komputasi utama. Penelitian selanjutnya menggunakan raspberry pi dan model SSD-MobilenetV2 untuk klasifikasi jenis kendaran. Sistem tersebut bertujuan untuk menghitung kendaraan yang masuk ke plabuhan merak berbasis *real-time* kemudian hasil klasifikasi dikirimkan ke MySQL. Hasil akurasi yang didapat menghasilkan rata-rata akurasi 99,405% untuk semua jenis kendaraan. Akurasi pengiriman data ke database menghasilkan rata-rata akurasi 84,661%. Hasil FPS yang didapat rata-rata 2.6 FPS [2].

Sensor PIR dan sensor buzzer digunakan sebagai alat masukan untuk sistem yang sedang berjalan, sedangkan kamera pi berfungsi untuk mengambil gambar dalam situasi tertentu. Sistem ini mengintegrasikan webserver di raspberry pi dan aplikasi Android yang terhubung melalui protokol IP. Sistem ini diberi nama sistem RO. Hasil dari eksperimen smarthome ini adalah sebuah solusi keamanan rumah untuk pemantauan jarak jauh melalui smartphone. Sistem RO mampu mendeteksi

objek hingga jarak 6 meter. Sensor PIR dan sensor buzzer memiliki waktu respons kurang dari 0,4 detik dan akurasi sensor mencapai 98,598, yang berkontribusi pada peningkatan kinerja alat keamanan rumah [3].

Sebagai hasilnya, dilakukan penelitian untuk membangun sistem keamanan rumah dengan menerapkan metode Waterfall. Sistem keamanan ini menggunakan sensor PIR untuk mendeteksi gerakan, setelah gerakan terdeteksi, Raspberry Pi memberi instruksi pada kamera untuk mengambil gambar dan mengirimkannya ke API Telegram, lalu ke Aplikasi Telegram. Ketika gambar terkirim, Raspberry Pi memicu speaker sebagai alarm peringatan. Pengguna juga memiliki kemampuan untuk memberikan perintah kepada Raspberry Pi untuk mengambil gambar atau video. Jarak maksimal yang dapat dideteksi oleh sensor PIR adalah 5 meter, sehingga jika objek berada lebih dari 5 meter jauhnya, sensor PIR tidak dapat mendeteksi pergerakan. Sensor PIR memiliki sudut sensitivitas yang bekerja pada kisaran sudut 45° hingga 135° [4].

Kamera Pi dipasang pada Raspberry Pi dengan sensor Passive Infrared dan lainnya. Kamera mengambil gambar di depan pintu, lalu pengenalan wajah dilakukan menggunakan Lokal Biner Pola (LBP), jika gambar cocok dengan anggota rumah atau orang yang dikenali, pintu terbuka; jika tidak, tetap terkunci. Gambar orang tak dikenal dikirim melalui email ke pemilik rumah. Sistem ini memberi informasi *real-time* tentang orang tak dikenal di dekat pintu, membantu pengguna mengambil Tindakan [5].

Penelitian sistem keamanan tempat parkir rumah berbasis Raspberry Pi 3 dengan layanan *email* gratis yaitu Gmail. Hasil pengujian sistem, deteksi gerakan dan pengambilan foto berhasil dengan nilai rata-rata 100% pada kondisi pagi, siang, dan malam. Sensor PIR memiliki jarak maksimum deteksi gerakan hingga 6 meter. Kontrol kamera untuk mengambil foto atau video menggunakan bot juga berhasil dengan nilai rata-rata keberhasilan 100%. Pengujian QoS, nilai rata-rata *delay* pada pengambilan foto dan video adalah 2,09 detik pada kondisi pagi, 2,41 detik pada siang hari, dan 2,30 detik pada malam hari. Pengujian *throughput* menunjukkan kecepatan rata-rata masing-masing sebesar 535.6 kB/s [6].

Penelitian berikut ini menggunakan sistem *Internet of Things (IoT)*. Tersedianya semakin banyak perangkat Internet of Things (IoT) di organisasi

publik dan pribadi juga memberikan solusi pengawasan pintar dan aman untuk pemantauan waktu nyata di ruang publik. sistem pengawasan kerumunan berbasis IoT yang menggunakan model pembelajaran mendalam untuk mendeteksi dan menghitung orang dengan menggunakan perspektif pandangan dari atas. Model Detektor Kotak Multibox Tunggal (SSD) dengan Mobilenetv2 sebagai jaringan dasarnya digunakan untuk mendeteksi manusia. Akurasi model deteksi ditingkatkan dengan pendekatan transfer *learning*. Dua garis virtual ditentukan untuk menghitung berapa banyak orang yang keluar dan masuk ke dalam adegan. Hasil menunjukkan bahwa transfer *learning* meningkatkan kinerja deteksi keseluruhan dari sistem dengan akurasi 95% [7].

BAB III

METODOLOGI PENELITIAN

3.1 Metode Penelitian

Penelitian ini merancang sistem *real-time* deteksi manusia dengan menggunakan Raspberry Pi. Data yang digunakan dalam penelitian ini berupa gambar manusia. Gambar tersebut diambil dari hasil video hasil rekaman menggunakan *Pi camera* dengan bantuan Raspberry Pi. Pengambilan data dan pengujian berlokasi di pekarangan rumah.

Berdasarkan tahapan-tahapan yang digunakan untuk menyelesaikan penelitian tersebut adalah sebagai berikut:

1. Melakukan studi literatur, Melakukan *research* tentang skripsi yang berkaitan dengan keamanan dan metode yang digunakan.
2. Melakukan pembuatan program, Proses pendeteksian manusia diperlukan *training* data. Sebelum proses *training* data diperlukan pengambilan data. Data untuk proses *training* diambil dari video yang ada di internet dan video yang diambil lokasi langsung. Setelah data terkumpul kemudian dilakukan proses *labeling* gambar dan *output* dari labeling gambar adalah sebuah *file* yang berisi letak koordinat objek dan nama dari objek tersebut. Selanjutnya proses training data akan dilakukan di Google Colaboratory. Output dari training data merupakan model SSD-MobilenetV2 dan SSD-Lite MobilenetV2. Program pendeteksian memiliki *trigger* yang akan mengenai objek yang dideteksi, sehingga jika ada yang mengenai *trigger* tersebut akan mengirimkan gambar dan teks ke Telegram
3. Melakukan pengujian, penelitian ini bertujuan membandingkan model SSD-MobilenetV2 dan SSD-Lite MobilenetV2. Model tersebut dilakukan perbandingan untuk mengetahui model yang sesuai dengan penelitian ini. Dari *research* yang telah dilakukan model tersebut banyak digunakan untuk perangkat *embedded* sistem seperti Raspberry Pi, Jetson Nano, Android atau *micro computer*. Model tersebut memiliki karakteristiknya sendiri seperti model SSD-MobilenetV2 memiliki *fps* yang kecil tetapi akurasinya tinggi dan SSD Lite-MobilenetV2 memiliki *fps* yang besar tetapi akurasinya tidak

sebaik model SSD-MobilenetV2.

4. Analisis hasil, Penelitian yang telah dilakukan mendapatkan model yang cocok digunakan untuk keamanan rumah secara *real-time.* Model yang tepat digunakan untuk penelitian ini tidak hanya memiliki *fps* dan akurasi yang tinggi dan juga dapat mengenai *trigger* dengan baik.
5. Mengambil kesimpulan, penelitian yang telah dilakukan menyimpulkan hasil akurasi dan fps yang dihasilkan oleh model SSD-MobilenetV2 dan SSD-Lite MobilenetV2, berserta dengan parameter yang mempengaruhi hasil akurasi dan fps.

3.2 Instrumen Penelitian

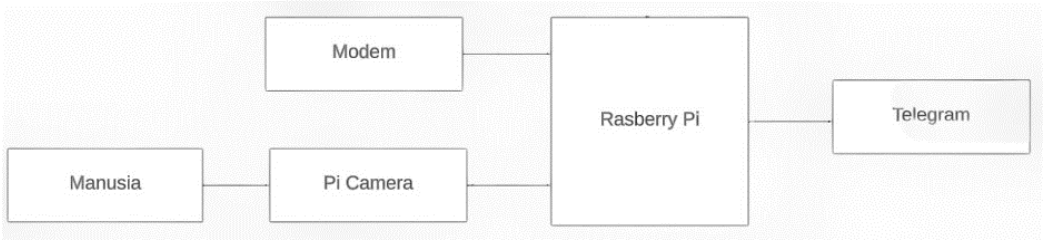
Alat yang digunakan dalam perancangan dan penelitian ini meliputi perangkat keras, perangkat lunak, dan database yang dapat mendukung dan mempermudah kinerja proses penelitian. Alat penelitian yang diperlukan disajikan pada Tabel 3.1.

Tabel 3.1 Instrumen Penelitian

No		Instrumen	Aplikasi
1.	*Hardware*	Raspberry Pi 4 RAM 2 GB	*Microcomputer* yang berfungsi untuk mengolah gambar, memproses gambar, dan mengirimkan data ke Telegram.
		Pi camera	Modul dari Raspberry Pi yang berfungsi untuk mengambil gambar.
		Modem Wi-Fi	Media *Hotspot.*
		Adaptor 5V	Sumber tegangan Raspberry Pi dan modem Wi-Fi.
		Micro Sd card	Sebagai *booTabel* OS Raspberry Pi dan penyimpanan data.
		Monitor	Sebagai *output* tampilan dari Raspberry Pi.
2.	*Software*	Python IDLE versi 3.2.3	Membuat *listing* program lalu menyusun dan mengunggah program pada Raspberry Pi.
		Telegram	Sebagai media pengiriman dan penampil gambar.

3.3 Diagram Blok Penelitian

Penelitian ini memiliki kerangka alur pendeteksian manusia. kerangka alur tersebut Digambar kan melalui diagram blok penelitian. Diagram blok penelitian dapat dilihat pada Gambar 3.1.

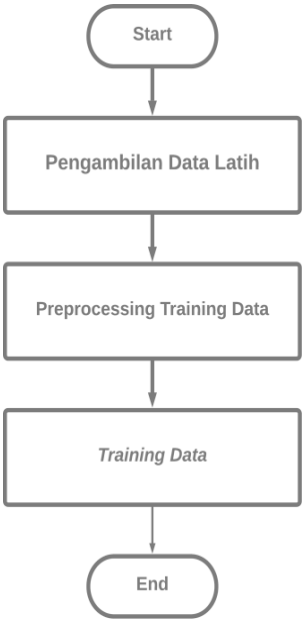


Gambar 3.1 Diagram Blok Penelitian

Diagram blok pada gambar 3.1 merupakan alur sistem pendeteksian jenis manusia menggunakan Raspberry Pi secara *real-time*. Menjelaskan alur proses penelitian ini mulai dari menangkap gambar manusia menggunakan Pi *camera* kemudian melakukan proses pendeteksian di Raspberry Pi dan hasil dari pendeteksian dikirimkan ke Telegram.

3.4 Pembuatan Model

Pembuatan model SSD-MobilenetV2 dan SSD Lite-MobilenetV2 langkah-langkah pembuatan model tersebut dapat dilihat pada Gambar 3.2.



Gambar 3.2 Proses Pembuatan Model

3.4.1 Pengambilan Data Latih

Data latih digunakan untuk membangun model SSD-MobilenetV2 dan SSD-Lite MobilenetV2. Data latih didapat dari pengambilan video lalu lalang manusia dari pagi hingga malam hari Menggunakan Raspberry Pi dan Pi *camera* yang berlokasi di rumah peneliti. Pengambilan data latih selama 30 hari dan setiap pengambilan merekam lalu lalang manusia kurang lebih 2 jam perhari. Data latih yang berupa video nantinya diambil gambar yang ada manusianya saja. Lokasi pengambilan untuk data latih dan pengujian dapat dilihat pada Gambar 3.3.

Gambar 3.3 Lokasi Pengambilan Dan Pengujian Data

3.4.2 *Preprocessing Training Data*

Pengolahan data latih bertujuan untuk membangun model SSD-MobilenetV2 dan SSD Lite-MobilenetV2. Data latih berupa gambar yang sudah diambil sebelumnya kemudian diproses sehingga menjadi model yang memiliki *extention* Tensorflow Lite. Berikut tahap-tahap mengolah data latih berikut ini.

1. Konfigurasi model SSD-MobilenetV2 dan SSD Lite-MobilenetV2 Konfigurasi model bertujuan untuk menentukan jumlah objek yang dideteksi, jumlah *batch size* dan lokasi *check point*.
2. Mengambil Gambar Manusia Dari Hasil Video Rekaman.

Data latih telah diambil sebelumnya berupa video orang yang melewati rumah peneliti melalui Raspberry Pi melalu *Pi camera*. Setelah data latih didapat, kemudian diambil gambar manusia dan disimpan.

3. Melakukan *Resizing*

   Gambar yang telah didapat dari hasil video rekaman orang yang melintas pada lokasi penelitian. *Resizing* dilakukan untuk mengubah rasio menjadi ukuran gambar yang lebih kecil sehingga efeknya adalah ukuran *file* gambarnya lebih kecil. Ukuran *file* gambar berpengaruh padaproses *training*, semakin kecil ukuran *file* maka semakin cepat proses *training* dan mempersingkat waktu untuk *upload* data *training* ke *google drive*.

4. Melakukan Pelabelan Gambar/Citra

   Proses selanjutnya adalah melakukan pelabelan gambar/citra atau *labeling*. *Labeling* gambar menggunakan aplikasi *Labeling* kemudian simpan dengan format Pascal/VOC supaya hasilnya berformat .XML. Proses anotasi gambar adalah dengan membuat kotak pada manusia. Pada proses ini menghasilkan *file* dengan format manusia.XML. Pada *file* manusia.XML berisi manusia yang sudah dinamai dan koordinat dari kotak *box* yang telah dilakukan sebelumnya.

3.4.3 *Training Data*

Proses *training* dilakukan di *google colab*, karna pada*google colab* terdapat super GPU yang mempercepat proses *training data*. *Output* yang dihasilkan setelah *training* adalah *file* yang berekstensi .pb. lalu dikonversi ke bentuk yang berekstensi tflite.

3.5 Konfigurasi Raspberry Pi

Raspberry Pi harus dikonfigurasi terlebih dahulu supaya dapat menjalan program yang dibuat. Konfigurasi Raspberry Pi dimulai dengan konfigurasi *Pi camera* pada Raspberry Pi. Berikut langkah-langkah konfigurasi Raspberry Pi:

1. Konfigurasi Raspberry Pi Untuk Dapat Melakukan Perekaman Video

   Pemasangan dan konfigurasi *Pi camera*, langkah selanjutnya adalah membuat Raspberry Pi supaya dapat melakukan perekaman menggunakan Raspberry Pi dengan *Pi camera*. Perekaman dengan Raspberry Pi dengan

menjalankan program yang dibuat oleh bahasa Python.

2. Konfigurasi Raspberry Pi Untuk Menginstall *Library* Yang Dibutuhkan

   Penelitian ini menggunakan *library* numpy, math, time, sys, opencv, argparse, datetime, dan tensorflow. Tujuan dari menginstall *library* tersebut adalah supaya program dapat melakukan deteksi manusia, sebelum *install* library maka harus *install* pip terlebih dahulu.

3.6 Alur Sistem

Penelitian ini terdapat alur sistem atau proses kerja. proses kerja tersebut meliputi menjalankan program, mendeteksi manusia dan Telegram. Alur sistem dapat dilihat pada Gambar 3.4 di bawah ini.

Gambar 3.4 *Flowchart* Alur Sistem

Berdasarkan *flowchart* alur sistem pada Gambar 3.4, maka dapat diketahui bahwa tahapan-tahapan yang digunakan untuk proses deteksi pada penelitian ini adalah sebagai berikut:

1. Menjalankan program yang telah dibuat yang dimana program tersebut terdapat model SSD-MobilenetV2 dan SSD Lite-MobilenetV2.
2. Mendeteksi manusia dengan bantuan *Pi camera* yang telah terkoneksi dengan program.

3. Telegram adalah *output* dari proses deteksi manusia.

3.7 Pengiriman Deteksi Ke Telegram

Pengiriman data gambar ke telegram dengan menggunakan *trigger* yang telah dibuat pada program. *Trigger* tersebut bekerja jika ada manusia yang melewati *trigger* tersebut, setelah itu mengirimkan data gambar ke telegram. Proses pengiriman gambar ke Telegram dapat dilihat pada Gambar 3.5.

Gambar 3.5 Proses Pengiriman ke Telegram

3.8 Tempat Dan Waktu Penelitian

Penelitian dilakukan di pekarangan rumah. Skripsi ini berlangsung dari bulan januari 2023.

BAB IV
HASIL DAN PEMBAHASAN

Penelitian ini membahas sistem perbandingan SSD-MobilenetV2 dengan SSD Lite-MobilenetV2 untuk pendeteksian manusia. Sistem tersebut menggunakan program yang dibuat dengan bahasa pemrograman Python dan *hardware* Raspberry Pi. Sistem ini akan mendeteksi manusia secara *real-time* dan *output* hasil dari pendeteksian dikirimkan ke telegram.

4.1 Hasil Pengambilan Data Latih

Pengambilan data latih menggunakan Raspberry Pi *Camera* yang berbentuk video dan video dari internet. Video yang telah didapat kemudian diambil gambar yang hanya terdapat objek manusia, gambar tersebut yang menjadi data latih. Pengambilan data latih dilakukan di rumah. Tabel 4.1 berisi jumlah gambar yang menjadi data latih.

Tabel 4.1 Jumlah Data Latih

Objek	**Jumlah Objek**
Person	964
Aan	126
Mamah	123
Edo	128
Total	1341

Pengambilan menggunakan Raspberry Pi *camera* diambil selama 3 hari. Hasil dari data latih mendapatkan jumlah data latih person sebanyak 964 objek, Aan sebanyak 126, Mamah sebanyak 123, Edo sebanyak 128 dan total data latih adalah 1341. Data Analisa tersebut akan diperlukan pada proses *training.*

4.2 Hasil Training Data

Data latih telah didapat kemudian selanjutnya diolah supaya dapat melakukan klasifikasi jenis manusia. Mengolah data latih mulai dari melakukan *resizing*, anotasi gambar, konversi xml ke csv, membuat *file record* dan membuat *labelmap*, setelah itu upload *file* ke Google Drive. Data yang sudah diolah kemudian

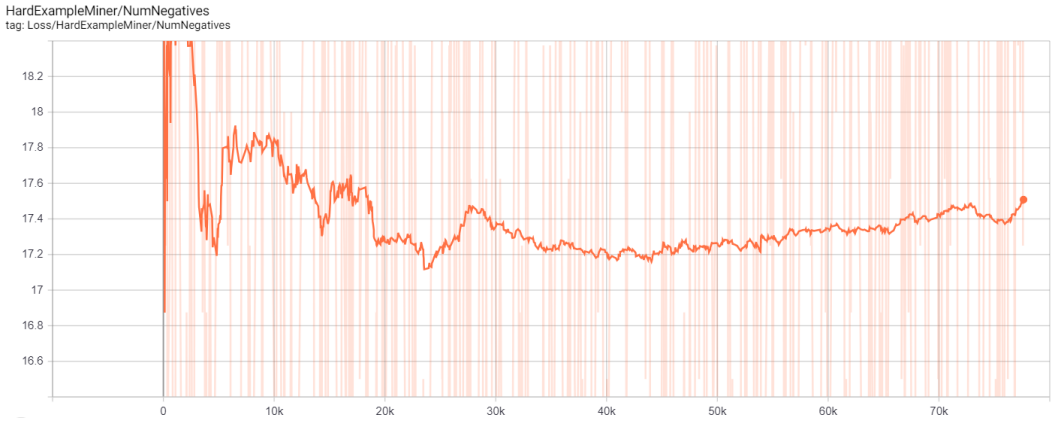
di *training* menggunakan bantuan dari Google Colaboratory karena disana terdapat super GPU yang membuat proses *training* menjadi cepat.

Proses *training* model SSD-MobilenetV2 membutuhkan waktu rata-rata 3 detik setiap 1 *step*. Penelitian ini membutuhkan 58246 *step* untuk SSD Lite-MobilenetV2 dan 76829 untuk SSD-MobilenetV2. Proses *training* dapat dilihat pada Gambar 4.1.

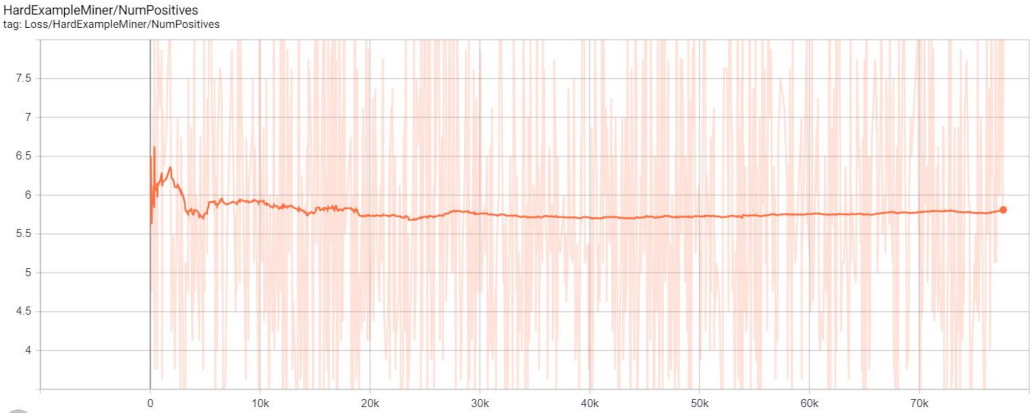
```
I0627 20:13:20.010804 12760 learning.py:507] global step 76797: loss = 2.1578 (15.369 sec/step)
INFO:tensorflow:global step 76798: loss = 1.8892 (13.265 sec/step)
I0627 20:13:33.275341 12760 learning.py:507] global step 76798: loss = 1.8892 (13.265 sec/step)
INFO:tensorflow:global step 76799: loss = 4.4745 (13.364 sec/step)
I0627 20:13:46.654924 12760 learning.py:507] global step 76799: loss = 4.4745 (13.364 sec/step)
INFO:tensorflow:global step 76800: loss = 1.8616 (13.274 sec/step)
I0627 20:13:59.928580 12760 learning.py:507] global step 76800: loss = 1.8616 (13.274 sec/step)
INFO:tensorflow:global step 76801: loss = 2.4588 (13.390 sec/step)
I0627 20:14:13.318986 12760 learning.py:507] global step 76801: loss = 2.4588 (13.390 sec/step)
INFO:tensorflow:global step 76802: loss = 2.2089 (13.421 sec/step)
I0627 20:14:26.744546 12760 learning.py:507] global step 76802: loss = 2.2089 (13.421 sec/step)
INFO:tensorflow:global step 76803: loss = 2.7703 (13.575 sec/step)
I0627 20:14:40.319196 12760 learning.py:507] global step 76803: loss = 2.7703 (13.575 sec/step)
INFO:tensorflow:global step 76804: loss = 3.3676 (13.165 sec/step)
I0627 20:14:53.484433 12760 learning.py:507] global step 76804: loss = 3.3676 (13.165 sec/step)
INFO:tensorflow:Recording summary at step 76804.
```

Gambar 4.1 Proses *Training*

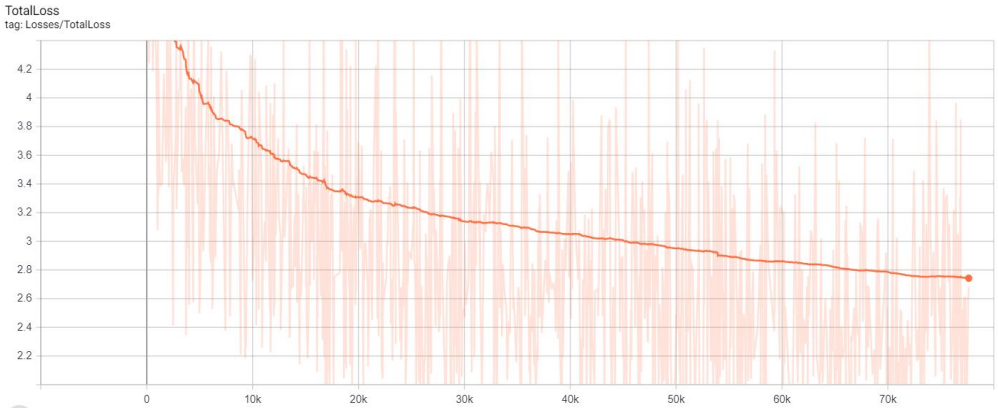
Proses *training* pada Gambar 4.1 menghasilkan Num*Negative*s, NumPositives, dan total *loss*. Proses *training* model SSD-MobilenetV2 sebanyak 77609 *step* menghasilkan Num*Negative*s dengan nilai 16 sampai 24 dan NumPositives dengan nilai 5 sampai 7, dapat dilihat pada Gambar 4.2 dan 4.3. Total *loss* yang dihasilkan dari *step* pertama hingga step terakhir adalah 5,8 hingga 1,8616, dapat dilihat pada Gambar 4.4.



Gambar 4.2 Grafik Nilai *NumNegatives*

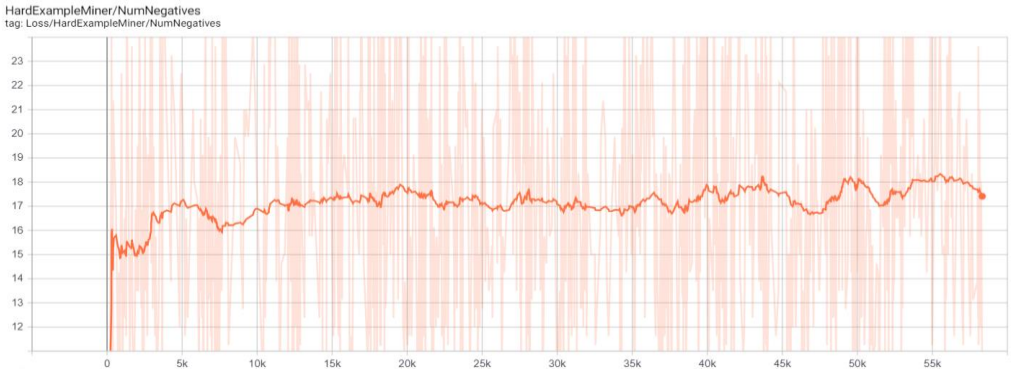


Gambar 4.3 Grafik Nilai *NumPositives*

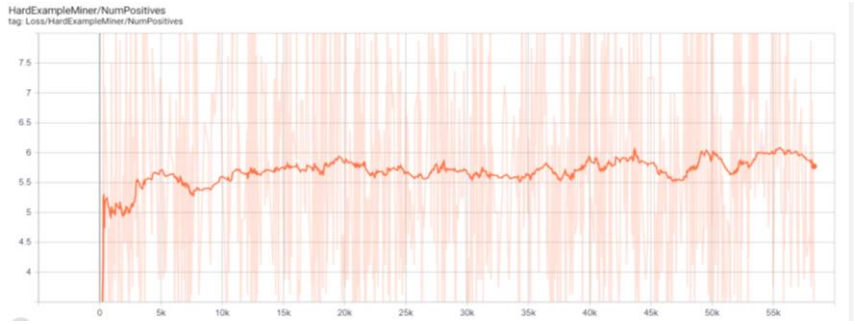


Gambar 4.4 Grafik *Total Loss*

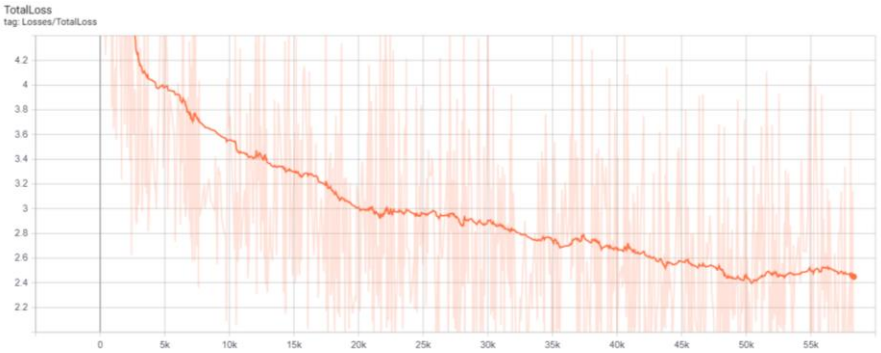
Proses *training* menghasilkan *NumNegatives*, *NumPositives*, dan total *loss*. Proses *training* model SSD Lite-MobilenetV2 sebanyak 58037 *step* menghasilkan *NumNegatives* dengan nilai 14 sampai 19 dan *NumPositives* dengan nilai 3 sampai 5, dapat dilihat pada Gambar 4.5 dan 4.6. Total *loss* yang dihasilkan dari *step* pertama hingga *step* terakhir adalah 6 hingga 1.854, dapat dilihat pada Gambar 4.7.



Gambar 4.5 Grafik Nilai *NumNegatives*



Gambar 4.6 Grafik Nilai *NumPositives*



Gambar 4.7 Grafik *Total Loss*

4.3 Klasifikasi Manusia

Proses klasifikasi dengan menangkap gambar yang diperoleh oleh Pi *camera* setelah diteruskan ke Raspberry Pi dengan menjalakan program yang telah dibuat. Program yang telah dibuat dilakukan proses *resize* terlebih dahulu ke 300x300 *pixel. Resize* dilakukan karna pada model SSD-MobilenetV2 hanya bisa melakukan proses pada ukuran pixel tersebut, setelah itu pada model SSD-MobilenetV2 dilakukan proses konvolusi, *activation ReLu, pooling, fully connected layer*. *Layer* konvolusi melakukan operasi konvolusi yaitu mengkombinasikan linier filter terhadap daerah lokal.

Bentuk *layer* ini adalah sebuah *filter* dengan P, L, T dengan *channel image* data yang dimasukkan ketiga *filter* ini bergeser keseluruh bagian gambar. Pergeserean tersebut melakukan operasi *dot* antara *input* dan nilai dari *filter* tersebut sehingga menghasilkan *output* yang disebut sebagai feature map. Proses selanjutnya adalah *activation* ReLu yang bertujuan supaya menghasilkan non-linearitas. Bekerja dengan mengubah nilai yang kurang dari 0 menjadi 0 sehingga

tidak ada nilai indeks gambar yang bernilai minus (-). *Pooling layer* menerima *output* dari konvolusi *layer*.

Pooling layer memiliki ukuran data citra direduksi. *Pooling layer* terdiri dari *filter* dengan ukuran tertentu dan *stride*/langkah kemudian bergeser keseluruh area *feature map*. Model SSD-MobilenetV2 menggunakan metode *mean pooling* yaitu membagi *output* konvolusi *layer* menjadi beberapa *grid*, kemudian setiap pergeseran *filter* akan mengambil nilai rata-rata dari setiap *grid*. *Output* pada *pooling layer* hasil masih berbentuk *multidimension array* sehingga harus melakukan *flattern* atau *reshape* menjadi sebuah vektor supaya bisa digunakan sebagai *input* dari *fully connected layer*. pada lapisan ini mendapat *input* dari proses sebelumnya untuk menentukan *filter* mana yang paling berkolerasi dengan kelas tertentu. Hasil dari proses tersebut akan mendeteksi manusia yang melintas secara *real-time*.

Objek manusia berhasil terdeteksi, maka muncul kotak hijau. Contoh hasil klasifikasi dapat dilihat pada Gambar 4.8. Memaksimalkan kinerja Raspberry Pi maka harus dilakukan *overclock* ke 2GHz dan harus diberikan pendingin supaya tidak terjadi *overheat*.

Gambar 4.8 Proses Klasifikasi Manusia

4.4 Mengirim Data Hasil Klasifikasi Ke Telegram

Mengirim data hasil klasifikasi dengan cara mengkoneksikan program Python dengan Telegram, sebelumnya harus membuat bot pada Telegram, setelah proses klasifikasi dilakukan, terdapat *trigger* yang berfungsi untuk mengirim hasil klasifikasi ke Telegram. *Trigger* tersebut dapat diatur sensitifitasnya sehingga

harus dikalibrasi terlebih dahulu supaya mendapakan hasil yang maksimal. Penelitian ini menggunakan 3 *trigger* untuk mendapatkan pendeteksian lebih akurat. Gambar 4.8 terdapat titik putih pada sisi kanan tengah kotak deteksi dan garis ungu pada *frame*. Titik putih tersebut akan menjadi *trigger* untuk mengirim data hasil klasifikasi, ketika titik putih mengenai garis ungu pada *frame*, maka akan langsung mengirimkan data ke Telegram. Hasil dari pengiriman data ke Telegram ketika terdeteksi *person* dapat dilihat pada Gambar 4.9.

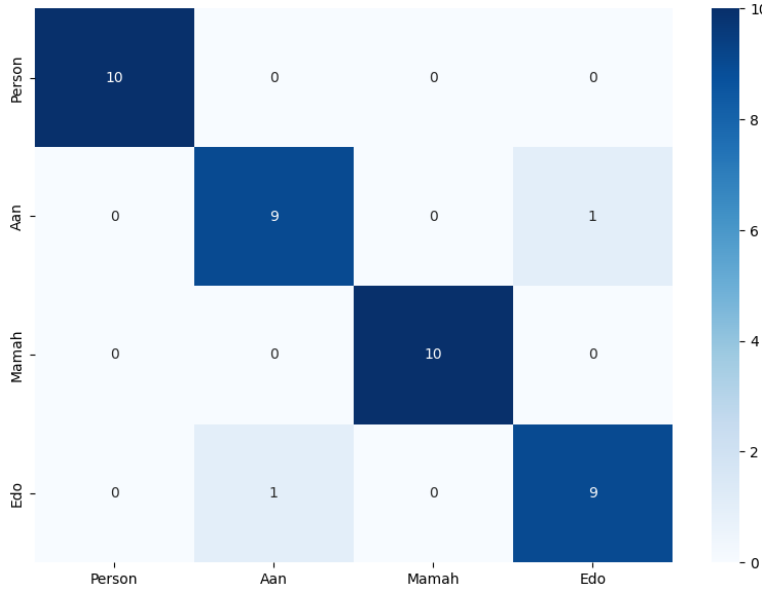
Gambar 4.9 Hasil Pengiriman ke Telegram

4.5 Pengujian Klasifikasi dan Pengiriman ke Telegram

Pengujian ini dipengaruhi dari jumlah *dataset* yang dimiliki, jumlah *step* yang diperoleh, pencahayaan, dan hasil resolusi yang dihasilkan oleh kamera. Proses pengujian ini menghitung tingkat akurasi deteksi manusia dan tingkat akurasi untuk mengirimkan data ke Telegram. Pengujian ini dilakukan sebanyak enam kali dengan tiga kali pengujuan SSD-MobilenetV2 dan tiga kali pengujian SSD Lite-MobilenetV2.

4.5.1 Pengujian Pertama

Pengujian dilakukan di rumah penguji pada tanggal 5 juni pukul 22.00 sampai dengan 23.59 dengan menggunakan 2 model yaitu SSD-MobilenetV2 dan SSD Lite-MobilenetV2 secara bergantian. Model pertama yaitu SSD-MobilenetV2 diuji pukul 22.00 sampai 22.59 dan model kedua yaitu SSD Lite-MobilenetV2 diuji pada pukul 23.00 sampai 23.59.



Gambar 4.10 *Confusion Matrix* Pada Pengujian Pertama

Tabel 4.2 Hasil Pengujian Hari Pertama SSD-MobilenetV2

Objek	Jumlah Aktual	TP	FP	FN	TN
Person	10	10	0	0	28
Aan	10	9	1	1	29
Mamah	10	10	0	0	28
Edo	10	9	1	1	29
Total	40	38	2	2	114

Tabel 4.2 merupakan hasil pengujian hari pertama dari SSD-MobilenetV2. Tabel menjelaskan bahwa TP merupakan *true* positif, FP merupakan *False* positif, FN merupakan *False negative* dan TN merupakan *True negative*. Objek yang digunakan dalam penelitian ini sebanyak 4 objek. Objek pertama terdapat nilai terdeteksi secara aktual sebanyak 10 kali, dengan nilai TP terdeteksi sebanyak 0, FP sebanyak 0, FN sebanyak 0 dan TN sebanyak 28. Objek kedua terdapat nilai

terdeteksi secara aktual sebanyak 10 kali, dengan nilai TP terdeteksi sebanyak 9, FP sebanyak 1, FN sebanyak 1 dan TN sebanyak 29. Objek ketiga terdapat nilai terdeteksi secara aktual sebanyak 10 kali, dengan nilai TP terdeteksi sebanyak 0, FP sebanyak 0, FN sebanyak 0 dan TN sebanyak 28. Objek keempat terdapat nilai terdeteksi secara aktual sebanyak 10 kali, dengan nilai TP terdeteksi sebanyak 9, FP sebanyak 1, FN sebanyak 1 dan TN sebanyak 29.

Berdasarkan Tabel 4.2 terdapat kesalahan pendeteksian pada objek kedua, objek ketiga dan objek keempat. Kesalahan pendeteksian terjadi pada objek kedua yaitu objek Aan yang terdeteksi sebagai objek Edo. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek kedua dan objek keempat. Nilai FN untuk objek kedua bernilai 1 dan nilai FP pada objek kempat bernilai 1. Kesalahan juga terjadi pada objek keempat yaitu objek Edo, dimana objek Edo terdeteksi sebagai objek kedua yaitu objek Aan. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek keempat dan objek kedua, dimana nilai FN untuk objek keempat menjadi bernilai 1 dan nilai FP pada objek kedua menjadi bernilai 1.

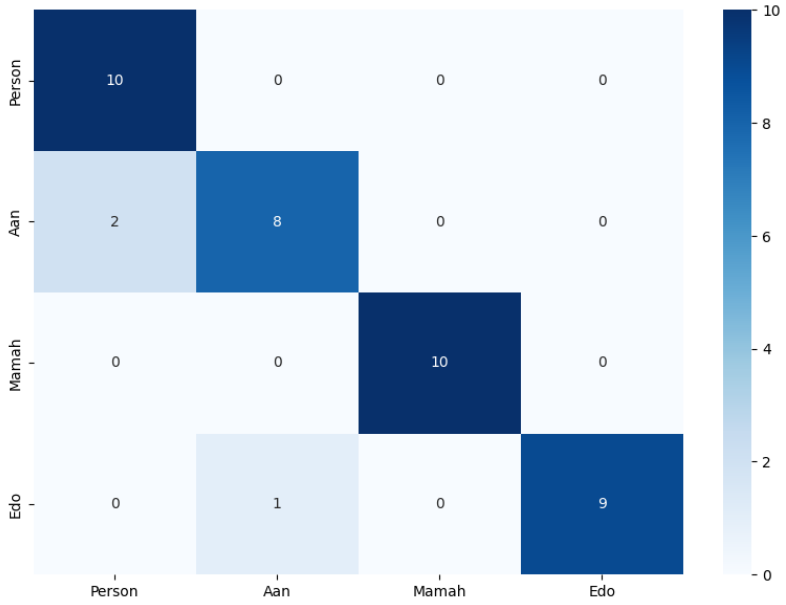
Data Tabel 4.2 dapat dicari nilai untuk dengan menggunakan persamaan 2.1 dan terdapat nilai *recall* sebesar 0,95, lalu persamaan 2.2 digunakan untuk mencari nilai *precision* sebesar 0,95, lalu persamaan 2.3 digunakan mencari nilai F-Score sebesar 0,94, lalu persamaan 2.4 digunakan untuk mencari akurasi sebesar 97,43%.

Tabel 4.3 Hasil Pengujian Hari Pertama Pengiriman ke Telegram

Objek	Jumlah Aktual	Berhasil Terkirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
person	10	10	0	0
Aan	10	9	1	0
Mamah	10	8	0	2
Edo	10	9	1	0
Total	40	36	2	2
Rata-rata		90%	5%	5%

Tabel 4.3 mendapatkan nilai rata-rata yang berhasil dikirim ke Telegram 90%, nilai rata-rata salah deteksi yang terkirim ke Telegram 5% dan Nilai rata-rata terdeteksi tidak terkirim ke Telegram 5%.

Pengujian dilakukan di rumah penguji pada tanggal 5 juni pukul 22.00 sampai dengan 23.59 dengan menggunakan 2 model yaitu SSD-MobilenetV2 dan SSD Lite-MobilenetV2 secara bergantian. Model pertama yaitu SSD-MobilenetV2 diuji pukul 22.00 sampai 22.59 dan model kedua yaitu SSD Lite-MobilenetV2 diuji pada pukul 23.00 sampai 23.59.



Gambar 4.11 *Confusion Matrix* Pada Pengujian Pertama

Tabel 4.4 Hasil Pengujian Hari Pertama SSD-MobilenetV2

Objek	Jumlah Aktual	TP	FP	FN	TN
person	10	10	0	2	27
Aan	10	8	2	1	29
Mamah	10	10	0	0	27
Edo	10	9	1	0	28
Total	40	37	3	3	111

Tabel 4.4 merupakan hasil pengujian hari pertama dari SSD Lite-MobilenetV2. Dalam Tabel tersebut dijelaskan bahwa TP merupakan *true* positif, FP merupakan *False* positif, FN merupakan *False negative* dan TN merupakan *True negative*. Objek yang digunakan dalam penelitian ini sebanyak 4 objek. Objek pertama terdapat nilai terdeteksi secara aktual sebanyak 10 kali, dengan nilai TP terdeteksi sebanyak 10, FP sebanyak 0, FN sebanyak 2 dan TN sebanyak 27. Objek kedua terdapat nilai terdeteksi secara aktual sebanyak 10 kali, dengan nilai TP

terdeteksi sebanyak 8, FP sebanyak 2, FN sebanyak 1 dan TN sebanyak 29. Objek ketiga terdapat nilai terdeteksi secara aktual sebanyak 10 kali, dengan nilai TP terdeteksi sebanyak 10, FP sebanyak 0, FN sebanyak 0 dan TN sebanyak 27. Objek keempat terdapat nilai terdeksi secara aktual sebanyak 10 kali, dengan nilai TP terdeteksi sebanyak 9, FP sebanyak 1, FN sebanyak 0 dan TN sebanyak 28.

Berdasarkan Tabel 4.4 terdapat kesalahan pendeteksian pada objek kedua yaitu objek Aan, objek ketiga yaitu objek Mamah dan objek keempat yaitu objek Edo. Kesalahan terjadi pada objek Aan yaitu terdeksi menjadi objek person. Nilai FN untuk objek pertama bernilai 2, nilai FP pada objek kedua bernilai 2, setelah itu ada terjadi kesahalan deteksi yaitu objek keempat yang terdeteksi menjadi objek kedua. Nilai FN untuk objek kedua yaitu bernilai 1, Nilai FP pada objek keempat yaitu bernilai 1.

Data Tabel 4.4 dapat dicari nilai untuk dengan menggunakan persamaan 2.1 dan terdapat nilai *recall* sebesar 0,925, lalu persamaan 2.2 digunakan untuk mencari nilai *precision* sebesar 0.925, lalu persamaan 2.3 digunakan mencari nilai *F-Score* sebesar 0.924, lalu persamaan 2.4 digunakan untuk mencari akurasi sebesar 96.1%.

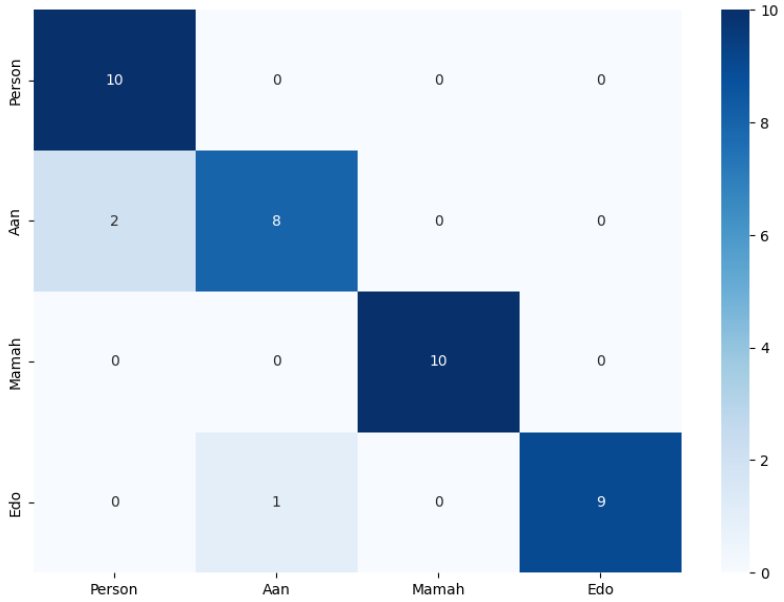
Tabel 4.5 Hasil Pengujian Hari Pertama Pengiriman ke Telegram

Objek	Jumlah Aktual	Berhasil Terkirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
person	10	10	0	0
Aan	10	8	2	0
Mamah	10	10	0	0
Edo	10	9	1	0
Total	40	37	3	0
Rata-rata		92,5%	7,5%	0%

Tabel 4.5 mendapatkan nilai rata-rata yang berhasil dikirim ke Telegram 92,5%, nilai rata-rata salah deteksi yang terkirim ke Telegram 7,5% dan Nilai rata-rata terdeteksi tidak terkirim ke Telegram 0%.

4.5.2 Pengujian Kedua

Pengujian dilakukan di rumah penguji pada tanggal 6 juni pukul 22.00 sampai dengan 23.59 dengan menggunakan 2 model yaitu SSD-MobilenetV2 dan SSD Lite-MobilenetV2 secara bergantian. Model pertama yaitu SSD-MobilenetV2 diuji pukul 22.00 sampai 22.59 dan model kedua yaitu SSD Lite-MobilenetV2 diuji pada pukul 23.00 sampai 23.59.



Gambar 4.12 *Confusion Matrix* Pada Pengujian Kedua

Tabel 4.6 Hasil Pengujian Hari kedua SSD-MobilenetV2

Objek	Jumlah Aktual	TP	FP	FN	TN
person	15	14	1	0	44
Aan	15	14	1	1	44
Mamah	15	15	0	0	43
Edo	15	15	0	1	43
Total	60	58	2	2	174

Tabel 4.6 merupakan hasil pengujian hari kedua dari SSD-MobilenetV2. Tabel tersebut menjelaskan bahwa TP merupakan *true* positif, FP merupakan *False* positif, FN merupakan *False negative* dan TN merupakan *True negative*. Objek yang digunakan dalam penelitian ini sebanyak 4 objek. Objek pertama terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 14,

FP sebanyak 1, FN sebanyak 0 dan TN sebanyak 44. Objek kedua terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 14, FP sebanyak 1, FN sebanyak 1 dan TN sebanyak 44. Objek ketiga terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 15, FP sebanyak 0, FN sebanyak 0 dan TN sebanyak 43. Objek keempat terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 0, FP sebanyak 0, FN sebanyak 1 dan TN sebanyak 43.

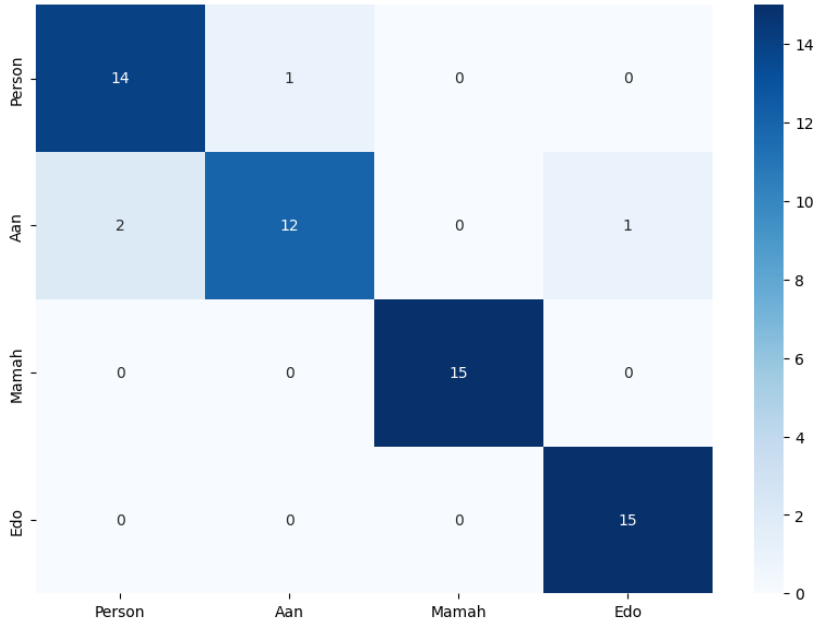
Berdasarkan Tabel 4.5 terdapat kesalahan pendeteksian pada objek pertama yaitu objek person. Kesalahan terjadi karena objek person terdeteksi sebagai objek kedua yaitu objek Aan. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek pertama dan objek. Nilai FP untuk objek pertama bernilai 1 dan nilai FN pada objek kedua bernilai 1, selanjutnya kesalahan pendeteksian pada objek kedua yaitu objek Aan. Kesalahan terjadi karena objek Aan terdeteksi sebagai objek keempat yaitu objek edo. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek kedua dan objek keempat. Nilai FN untuk objek kedua bernilai 1 dan nilai FP pada objek keempat bernilai 1.

Data Tabel 4.6 dapat dicari nilai untuk dengan menggunakan persamaan 2.1 dan terdapat nilai *recall* sebesar 0,96 lalu persamaan 2.2 digunakan untuk mencari nilai *precision* sebesar 0,96, lalu persamaan 2.3 digunakan mencari nilai F-Score sebesar 0,95, lalu persamaan 2.4 digunakan untuk mencari akurasi sebesar 98,30%.

Tabel 4.7 Hasil Pengujian Hari kedua Pengiriman ke Telegram

Objek	Jumlah Aktual	Berhasil Terkirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
person	15	13	1	1
Aan	15	14	1	0
Mamah	15	14	0	1
Edo	15	12	0	3
Total	60	53	2	5
Rata-rata		92,5%	3,33%	8,3%

Tabel 4.7 merupakan hasil pengujian hari kedua pengiriman ke Telegram pada model SSD-MobilenetV2. Nilai rata-rata yang berhasil dikirim ke Telegram 92,5%, nilai rata-rata salah deteksi yang terkirim ke Telegram 3,33% dan Nilai rata-rata terdeteksi tidak terkirim ke Telegram 8,3%.



Gambar 4.13 *Confusion Matrix* Pada Pengujian Kedua

Tabel 4.8 Hasil Pengujian Hari Kedua SSD Lite-MobilenetV2

Objek	Jumlah Aktual	TP	FP	FN	TN
person	15	14	1	2	42
Aan	15	12	3	1	44
Mamah	15	15	0	0	41
Edo	15	15	0	1	41
Total	60	56	4	4	168

Tabel 4.8 merupakan hasil pengujian hari kedua dari SSD Lite-MobilenetV2. Tabel tersebut menjelaskan bahwa TP merupakan *true* positif, FP merupakan *False* positif, FN merupakan *False negative* dan TN merupakan *True negative*. Objek yang digunakan dalam penelitian ini sebanyak 4 objek. Objek pertama terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 14, FP sebanyak 1, FN sebanyak 2 dan TN sebanyak 42. Objek kedua terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 12, FP sebanyak 3, FN sebanyak 1 dan TN sebanyak 44. Objek

ketiga terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 15, FP sebanyak 0, FN sebanyak 0 dan TN sebanyak 41. Objek keempat terdapat nilai terdeteksi secara aktual sebanyak 15 kali, dengan nilai TP terdeteksi sebanyak 15, FP sebanyak 0, FN sebanyak 1 dan TN sebanyak 41.

Berdasarkan Tabel 4.8 terdapat kesalahan pendeteksian pada objek pertama yaitu objek person. Kesalahan terjadi karena objek person terdeteksi sebagai objek kedua yaitu objek Aan. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek pertama dan objek kedua. Nilai FP untuk objek pertama bernilai 1 dan nilai FN pada objek kedua bernilai 2, selanjutnya kesalahan pendeteksian pada objek kedua yaitu objek Aan. kesalahan terjadi karena objek Aan terdeteksi sebagai objek pertama yaitu objek person dan objek keempat yaitu objek Edo. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek pertama, objek kedua dan objek keempat. Nilai FN untuk objek pertama bernilai 2, nilai FP pada objek kedua bernilai 3 dan nilai FN pada objek keempat yiatu bernilai 1.

Data Tabel 4.8 dapat dicari nilai untuk dengan menggunakan persamaan 2.1 dan terdapat nilai *recall* sebesar 0,93 lalu persamaan 2.2 digunakan untuk mencari nilai *precision* sebesar 0,93, lalu persamaan 2.3 digunakan mencari nilai F-Score sebesar 0,92, lalu persamaan 2.4 digunakan untuk mencari akurasi sebesar 96,55%.

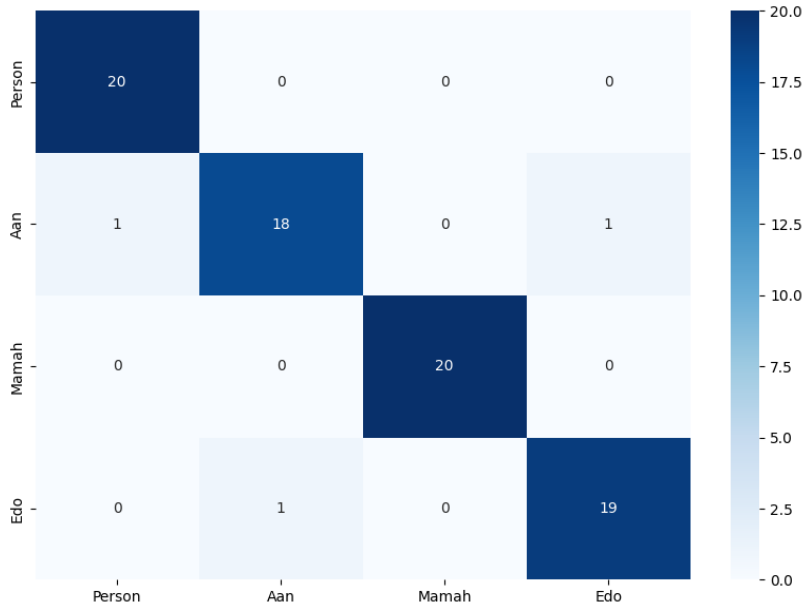
Tabel 4.9 Hasil Pengujian Hari kedua Pengiriman ke Telegram

Objek	Jumlah Aktual	Berhasil Terkirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
person	15	14	1	0
Aan	15	12	3	0
Mamah	15	15	0	0
Edo	15	15	0	0
Total	60	56	4	0
Rata-rata		93,3%	6,6%	0%

Tabel 4.9 merupakan hasil pengujian hari keduda pengiriman ke Telegram pada model SSD Lite-MobilenetV2. Nilai rata-rata yang berhasil dikirim ke Telegram 93,3%, nilai rata-rata salah deteksi yang terkirim ke Telegram 6,6% dan Nilai rata-rata terdeteksi tidak terkirim ke Telegram 0%.

4.5.3 Pengujian Ketiga

Pengujian dilakukan di rumah penguji pada tanggal 7 juni pukul 22.00 sampai dengan 23.59 dengan menggunakan 2 model yaitu SSD-MobilenetV2 dan SSD Lite-MobilenetV2 secara bergantian. Model pertama yaitu SSD-MobilenetV2 diuji pukul 22.00 sampai 22.59 dan model kedua yaitu SSD Lite-MobilenetV2 diuji pada pukul 23.00 sampai 23.59.



Gambar 4.14 *Confusion Matrix* Pada Pengujian ketiga

Tabel 4.10 Hasil Pengujian Hari ketiga SSD-MobilenetV2

Objek	Jumlah Aktual	TP	FP	FN	TN
person	20	20	0	1	57
Aan	20	18	2	1	59
Mamah	20	20	0	0	57
Edo	20	19	1	1	58
Total	80	77	3	3	231

Tabel 4.10 merupakan hasil pengujian hari ketiga dari SSD-MobilenetV2. Dalam Tabel tersebut dijelaskan bahwa TP merupakan *true* positif, FP merupakan *False* positif, FN merupakan *False negative* dan TN merupakan *True negative*. Objek yang digunakan dalam penelitian ini sebanyak 4 objek. Objek pertama terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi

sebanyak 20, FP sebanyak 0, FN sebanyak 1 dan TN sebanyak 57. Objek kedua terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi sebanyak 18, FP sebanyak 2, FN sebanyak 0 dan TN sebanyak 59. Objek ketiga terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi sebanyak 0, FP sebanyak 0, FN sebanyak 0 dan TN sebanyak 57. Objek keempat terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi sebanyak 19, FP sebanyak 1, FN sebanyak 1 dan TN sebanyak 58.

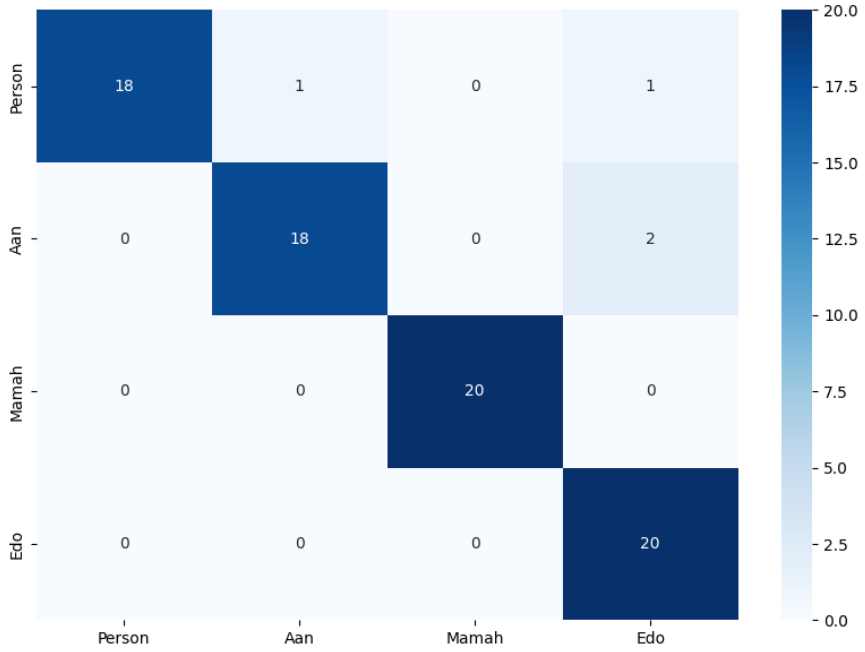
Berdasarkan Tabel 4.10 terdapat kesalahan pendeteksian pada objek pertama yaitu objek person. Kesalahan terjadi karena objek person terdeteksi sebagai objek kedua yaitu objek Aan dan objek keempat yaitu objek Edo. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek pertama, objek kedua dan objek keempat. Nilai FP untuk objek pertama bernilai 2, nilai FN pada objek kedua bernilai 1 dan nilai FN pada objek keempat yiatu bernilai 1. Selanjutnya kesalahan pendeteksian pada objek kedua yaitu objek Aan. kesalahan terjadi karena objek Aan terdeteksi sebagai objek pertama yaitu objek person dan objek keempat yaitu objek Edo. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek pertama, objek kedua dan objek keempat. Nilai FN untuk objek pertama bernilai 1, nilai FP pada objek kedua bernilai 2 dan nilai FN pada objek keempat yiatu bernilai 1.

Dari data Tabel 4.10 dapat dicari nilai dengan menggunakan persamaan 2.1 dan terdapat nilai *recall* sebesar 0,962 lalu persamaan 2.2 digunakan untuk mencari nilai *precision* sebesar 0,962, lalu persamaan 2.3 digunakan mencari nilai F-Score sebesar 0,961, lalu persamaan 2.4 digunakan untuk mencari akurasi sebesar 98,09%.

Tabel 4.11 Hasil Pengujian Hari Ketiga Pengiriman ke Telegram

Objek	Jumlah Aktual	Berhasil Terkirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
Person	20	19	0	1
Aan	20	18	2	0
Mamah	20	20	0	0
Edo	20	19	1	0
Total	80	77	3	1
Rata-rata		95%	3,75%	1,25%

Tabel 4.11 merupakan hasil pengujian hari ketiga pengiriman ke Telegram pada model SSD-MobilenetV2. Nilai rata-rata yang berhasil dikirim ke Telegram 95%, nilai rata-rata salah deteksi yang terkirim ke Telegram 3,75% dan Nilai rata-rata terdeteksi tidak terkirim ke Telegram 1,25%.



Gambar 4.15 *Confusion Matrix* Pada Pengujian ketiga

Tabel 4.12 Hasil Pengujian Hari Ketiga SSD Lite-MobilenetV2

Objek	Jumlah Aktual	TP	FP	FN	TN
Person	20	18	2	0	58
Aan	20	18	2	1	58
Mamah	20	20	0	0	56
Edo	20	20	0	3	56
Total	80	76	4	4	240

Tabel 4.12 merupakan hasil pengujian hari ketiga dari SSD Lite-MobilenetV2. Dalam Tabel tersebut dijelaskan bahwa TP merupakan *true* positif, FP merupakan *False* positif, FN merupakan *False negative* dan TN merupakan *True negative*. Objek yang digunakan dalam penelitian ini sebanyak 4 objek. Objek pertama terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi sebanyak 18, FP sebanyak 2, FN sebanyak 0 dan TN sebanyak 58. Objek

kedua terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi sebanyak 18, FP sebanyak 2, FN sebanyak 1 dan TN sebanyak 58. Objek ketiga terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi sebanyak 20, FP sebanyak 0, FN sebanyak 0 dan TN sebanyak 56. Objek keempat terdapat nilai terdeteksi secara aktual sebanyak 20 kali, dengan nilai TP terdeteksi sebanyak 20, FP sebanyak 0, FN sebanyak 3 dan TN sebanyak 56.

Berdasarkan Tabel 4.12 terdapat kesalahan pendeteksian pada objek kedua yaitu objek Aan. Kesalahan terjadi karena objek Aan terdeteksi sebagai objek pertama yaitu objek person dan objek keempat yaitu objek Edo. Kesalahan tersebut dapat dilihat pada nilai FP dan FN antara objek pertama, objek kedua dan objek keempat. Nilai FP untuk objek pertama bernilai 1, nilai FN pada objek kedua bernilai 1 dan nilai FP pada objek keempat yiatu bernilai 1

Data Tabel 4.12 dapat dicari nilai untuk dengan menggunakan persamaan 2.1 dan terdapat nilai *recall* sebesar 0,95, lalu persamaan 2.2 digunakan untuk mencari nilai *precision* sebesar 0,95, lalu persamaan 2.3 digunakan mencari nilai F-Score sebesar 0,95, lalu persamaan 2.4 digunakan untuk mencari akurasi sebesar 97,53%.

Tabel 4.13 Hasil Pengujian Hari Ketiga Pengiriman ke Telegram

Objek	Jumlah Aktual	Berhasil Terkirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
person	20	18	2	0
Aan	20	18	2	0
Mamah	20	20	0	0
Edo	20	20	0	0
Total	80	76	4	0
Rata-rata		95%	5%	0%

Tabel 4.13 merupakan hasil pengujian hari ketiga pengiriman ke Telegram pada model SSD-MobilenetV2. Nilai rata-rata yang berhasil dikirim ke Telegram 95%, nilai rata-rata salah deteksi yang terkirim ke Telegram 5% dan Nilai rata-rata terdeteksi tidak terkirim ke Telegram 0%.

4.6 Hasil Seluruh Pengujian

Bagian ini akan dilakukan perbandingan hasil pada masing-masing percobaan yang telah dilakukan. Hasil yang didapatkan dari masing-masing percobaan yang telah dilakukan pada Tabel 4.14 sampai Tabel 4.17.

Tabel 4.14 Hasil Seluruh Pengujian SSD-MobilenetV2

	Pengujian			
	1	**2**	**3**	**Rata-rata**
Recall	0,95	0,96	0,962	0,957
Precision	0,95	0,96	0,962	0,957
F-Score	0,94	0,95	0,961	0,951
Akurasi	97,43%	98,30%	98,09%	97,94%
FPS	2,5	2,6	2,6	2,5

Tabel 4.14 hasil percobaan yang telah dilakukan pada model SSD-MobilenetV2, rata-rata akurasi yang diperoleh adalah 97,35%. Rata-rata FPS yang didapat saat pengujian adalah 2,5. Akurasi tertinggi diperoleh pada pengujian ketiga dengan tingkat akurasi model untuk deteksi manusia sebesar 98,09%.

Tabel 4.15 Hasil Keseluruhan Pengiriman Ke Telegram SSD-MobilenetV2

	Berhasil Kirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
Percobaan Pertama	90%	5%	5%
Percoobaan Kedua	92,5%	3,33%	8,33%
Percobaan Ketiga	95%	3,75%	1,25%
Rata-rata	92,5%	4,02%	4,86%

Tabel 4.15 merupakan hasil dari keseluruhan percobaan pada SSD-MobilenetV2. Pengiriman ke Telegram yang berhasil terkirim mendapatkan rata-rata pengiriman adalah 92,5%, Salah deteksi terkirim ke Telegram mendapatkan rata-rata pengiriman 4,02% dan tidak terkirim ke telegram 4,86%.

Sensitifitas yang tepat pada penelitian ini dapat diketahui dengan adanya cahaya supaya mendapatkan hasil yang maksimal. Alasan hasil pengujian ketiga pendeteksiannya lebih maksimal dikarenakan terdapat pencahayaan yang sangat mendukung dibandingkan percobaan pertama dan percobaan kedua yang kurang maksimal pencahayaannya.

Tabel 4.16 Hasil Seluruh Pengujian SSD Lite-MobilenetV2

	Pengujian			
	1	**2**	**3**	**Rata-rata**
Recall	0,925	0,93	0,95	0,935
Precision	0,925	0,93	0,95	0,935
F-Score	0,924	0,92	0,95	0,931
Akurasi	96,1%	96,55%	97,53%	96,72%
FPS	5,6	5,6	5,8	5,6

Tabel 4.16 hasil percobaan yang telah dilakukan pada model SSD Lite-MobilenetV2, rata-rata akurasi yang diperoleh adalah 96,72%, Rata-rata FPS yang didapat saat pengujian adalah 5,6 dan akurasi tertinggi diperoleh pada pengujian ketiga dengan tingkat akurasi model untuk deteksi manusia sebesar 97,53%.

Tabel 4.17 Hasil Keseluruhan Pengiriman Ke Telegram SSD Lite-MobilenetV2

	Berhasil Kirim ke Telegram	Salah Deteksi Terkirim ke Telegram	Terdeteksi Tidak Terkirim ke Telegram
Percobaan Pertama	92,5%	7,5%	0%
Percoobaan Kedua	93,3%	6,6%	0%
Percobaan Ketiga	95%	5%	0%
Rata-rata	93,27%	6,3%	0%

Tabel 4.17 merupakan hasil dari keseluruhan percobaan pada SSD-MobilenetV2. Pengiriman ke Telegram yang berhasil terkirim mendapatkan rata-rata pengiriman adalah 93,27%, Salah deteksi terkirim ke Telegram mendapatkan rata-rata pengiriman 6,3% dan tidak terkirim ke telegram 0%.

Sensitifitas yang tepat pada penelitian ini dapat diketahui dengan adanya cahaya supaya mendapatkan hasil yang maksimal. Alasan hasil pengujian ketiga pendeteksiannya lebih maksimal dikarenakan terdapat pencahayaan yang sangat mendukung dibandingkan percobaan pertama dan percobaan kedua yang kurang maksimal pencahayaannya.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan mengenai perbandingan SSD-MobilenetV2 dengan SSD Lite-MobilenetV2 menggunakan raspberry pi didapatkan kesimpulan sebagai berikut.

1. Proses pendeteksian manusia menggunakan Raspberry Pi secara *real-time* dengan menggunakan model SSD-MobilnetV2 dan SSD LITE-MobilenetV2 kemudian menggunakan framework Tensorflow Lite untuk membantu menyelesaikan perhitungan yang terdapat di model SSD-MobilenetV2 dan model SSD Lite-MobilenetV2.
2. Proses pengiriman hasil data klasifikasi ke Telegram dengan cara menggunakan *trigger* yang telah diletakan pada ujung *box* deteksi. Data yang dikirim ke Telegram yaitu berupa foto dan kata peringatan bila terdeteksinya objek person.
3. Tingkat akurasi Raspberry Pi untuk mengklasifikasi deteksi manusia sebanyak 3 kali dengan menggunakan model SSD-MobilenetV2 dan SSD LITE-MobilenetV2 yang mendapatkan akurasi pada model SSD-MobilenetV2 rata-rata 97,35% dan model SSD Lite-MobilenetV2 mendapatakan rata-rata akurasi 96,72%. Model SSD-MobilenetV2 mendapatkan rata-rata FPS 2,5 dan SSD Lite-MobilenetV2 yang mendapatkan rata-rata FPS 5,6. Penelitian ini model yang baik digunakan yaitu Model SSD Lite-MobilenetV2 karena mempunyai nilai FPS yang baik dan nilai akurasinya tidak jauh dari SSD-mobilenetV2.
4. Pendeteksian tidak dapat melakukan klasifikasi manusia apabila pencahayaan kurang dan karena resolusi webcam tidak memiliki sensor *infrared*.

5.2 Saran

Saran untuk pengembangan penelitian selanjutnya, antara lain:

1. Penelitian ini menggunakan data latih sebanyak 1341 gambar, untuk menghasilkan keakuratan yang lebih baik lagi dibutuhkan data latih yang

lebih banyak lagi dan memiliki jumlah data yang sama disetiap pendeteksian objek manusia.

2. Pemaksimalan kerja Raspberry Pi maka diperlukan *overclock* ke 2GHz dan memberikan pendingin supaya tidak terjadi *overheat*. Efek dari *overclock* adalah bertambahnya FPS.
3. Penelitian ini tidak dapat melakukan klasifikasi objek manusia dengan baik bila kurangnya pencahayan dikarenakan kamera *webcam* tidak memiliki sensor *infrared*. Penulis berharap supaya peneliti selanjutnya dapat melakukan klasifikasi manusia dengan pencahayaan dengan baik dan memiliki kamera dengan sensor *infrared*.

DAFTAR PUSTAKA

[1] Badan Pusat Statistik. *Statistik Kriminal 2022* . Jakarta. 2022.

[2] Santoso, M. I., & C, Indrajaya. Pengolahan Citra Untuk Klasifikasi Jenis Kendaraan Menggunakan Raspberry Pi Secara Real Time. 2020 Vol.1 No.1

[3] Saraswati, I., R, Oktavfazrin., & R, Fahrizal. *Prototipe of home security system using passive infra red and vibration senso based android.* 2019

[4] Sumboro, B,. Sutariyani., & R, I, Utomo. "Sistem Keamanan Rumah Berbasis Raspberry Pi dan Menggunakan Sensor PIR" 2020 Vol.26 No.1

[5] Dhobale, M,J., R, Y., Biradar., R, R, Pawar., & S, A, Awatade,. *Smart Home Security System using Iot, Face Recognition and Raspberry Pi.* 2020

[6] Dompeipen, T, A. & S, R, U, A, Sompie. Penerapan Computer Vision Untuk Pendeteksian Dan Penghitung Jumlah Manusia. 2021 Jurnal Teknik Informatika vol.15 no. 4

[7] Ahmed, I., M, Ahmad., A, Ahmad., & G, Jeon. "IoT-based crowd monitoring system: Using SSD with transfer learning" 2021 Vol 93.

[8] Thohari, A. N. A., & R, Adhitama. *Real-Time Object Detection For Wayang Punakawan Identification Using Deep Learning.* INFOTE (Informatics, Telecomunication, and Electronics). 2019. Vol. 11, No. 4.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation" arXiv preprint arXiv:1801.04381, 2018.]. Arsitektur SSD Lite bisa dilihat pada gambar 2.5.

[10] Hanugra Aulia Sidharta, S.T., M.MT. *Introduction to Open CV*. 2017. Tersedia dari: https://binus.ac.id/malang/2017/10/introduction-to-open-cv. [URL dikunjungi pada 4 Desember 2022].

[11] Nayyar, A., & V, Puri. Raspberry Pi- A Small, Powerful, Cost Effective and Efficient Form Factor Computer: A Review. 2015 Vol.5 no.12

[12] Jatmiko, W., & P, Mursanto. *RTOS* (*Real Time Operating System)*. 2015, Depok: Universitas Indonesia.

[13] Raspberry Pi Foundation. Camera Module. Tersedia dari: https://www.raspberrypi.com/documentation/computers/camera_software.html#introducing-the-raspberry-pi-cameras [URL dikunjungi pada 10 Desember 2022].

[14] Fahana, J. F., & Ridho, F. (2018). Pemanfaatan Telegram Sebagai Notifikasi Serangan untuk Keperluan Forensik Jaringan. JOM FISIP, 5(1), 1–11.].

[15] Google. *Colaboratory: Frequently Asked Questions.* Tersedia dari https://research.google.com/colaboratory/faq.html. [URL dikunjungi pada 20 Desember 2022].

[16] Tensorflow: *Model optimization*. 2021. Tersedia dari: https://www.tensorflow.org/lite/performance/model_optimization. [URL dikunjungi pada 4 Desember 2022].

[17] Zamidra, E: *Cara Mudah Membuat Jaringan Wireless*. 2014, Jakarta : PT. Gramedia.

[18] Y. Yao, Z. Qiu and M. Zhong, "Application of improved MobileNet-SSD on underwater sea cucumber detection robot," 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 2019, pp. 402-407, doi: 10.1109/IAEAC47372.2019.8997970.

[19] Sindy, F. *Pendeteksian Objek Manusia Secara Real Time Dengan Metode Mobilenet-SSD Menggunakan Movidius Neural Compute Stick Pada Raspberry Pi*. 2019, Universitas Sumatera Utara.

[20] Syahrudin, S., & T, Kurniawan. *Input dan Output Pada Bahasa Pemrograman Python*. Jurnal Dasar Pemograman Python STMIK, Juni 2018.

[21] Carneiro, T., et al. "*Performance Analysis of Google Colaboratory as aTool for Accelerating Deep Learning Applications"*. IEEE Access. 2018. Vol. 6, pp. 61677-61685.

[22] Fitriansyah, F. & Aryadillah. "Penggunaan Telegram Sebagai Media Komunikasi Dalam Pembelajaran Online". *Jurnal Humaniora*, Vol. 20, no. 2, September 2020.

[23] Bramer, Max: " *Principles of Data Mining"*. Springer Verlag London Limited, 2007.

LAMPIRAN

Program Merekam Menggunakan Raspberry Pi dan Pi Camera

```
 import numpy as npimport os
import cv2
 filename = 'video.avi' frames_per_second = 24.0res = '720p'
 # Set resolution for the video capture
 # Function adapted from https://kirr.co/0l6qmhdef change_res(cap,
 width, height):
 cap.set(3, width) cap.set(4, height)
 # Standard Video Dimensions Sizes STD_DIMENSIONS =  {
 "480p": (640, 480),
720p": (1280, 720),
 "1080p": (1920, 1080),
 "4k": (3840, 2160),
}

 # grab resolution dimensions and set video capture to it.def
 get_dims(cap, res='1080p'):
 width, height = STD_DIMENSIONS["720p"]if res in STD_DIMENSIONS:
 width,height = STD_DIMENSIONS[res]## change the current caputre
 device ## to the resulting resolution change_res(cap, width,
 height)
 return width, height
  # Video Encoding, might require additional installs# Types of
  Codes: http://www.fourcc.org/codecs.php VIDEO_TYPE = {
 'avi': cv2.VideoWriter_fourcc(*'XVID'), #'mp4':
 cv2.VideoWriter_fourcc(*'H264'),'mp4':
 cv2.VideoWriter_fourcc(*'XVID'),
 }
 def get_video_type(filename):
 filename,ext = os.path.splitext(filename)if ext in VIDEO_TYPE:
 return VIDEO_TYPE[ext]return VIDEO_TYPE['avi']
 cap = cv2.VideoCapture(0)
 out = cv2.VideoWriter(filename, get_video_type(filename), 25,
 get_dims(cap, res))

 while True:
 ret, frame = cap.read() out.write(frame) cv2.imshow('frame',frame)
 if cv2.waitKey(1) & 0xFF == ord('q'):break
 cap.release() out.release() cv2.destroyAllWindows()
```

Program Konversi Video Ke Gambar

```
Import cv2
import os
def video_to_frames(video, path_output_dir): vidcap =
cv2.VideoCapture(video)
count = 0
while vidcap.isOpened():
success, image = vidcap.read() if cv2.waitKey(1)& 0xFF == 32:
cv2.imwrite(os.path.join(path_output_dir, 'kendaraan%d.png') %
count, image)
count += 1 else:
break cv2.destroyAllWindows() vidcap.release()
```

Normalisasi Gambar

```
from PIL import Image import os

import argparse


def rescale_images(directory, size): for img in
os.listdir(directory):

im = Image.open(directory+img)

im_resized = im.resize(size, Image.ANTIALIAS)
im_resized.save(directory+img)


if name == ' main ':

parser = rgparse.ArgumentParser(description="Rescaleimages")
parser.add_argument('-d', '--directory', type=str, required=True,
help='Directory containing the images') parser.add_argument('-s',
'--size', type=int, nargs=2, required=True, metavar=('width',
'height'), help='Image size')

args = parser.parse_args() rescale_images(args.directory,
args.size)
```

Program deteksi Manusia

```
import numpy as np
import math
import time
import sys
import cv2
import argparse
import datetime
import telepot

from tensorflow.lite.python.interpreter import Interpreter

fps = ""
detectfps = ""
framecount = 0
detectframecount = 0
time1 = 0
time2 = 0
EntranceCounter = 0
EntranceCounter2 = 0

def CheckEntranceLineCrossing(ymax, CoorXEntranceLine):
    AbsDistance = abs(ymax - CoorXEntranceLine)

    if AbsDistance <= 9:
        return 1
    else:
        return 0

def CheckEntranceLineCrossing2(ymax, CoorXEntranceLine2):
    AbsDistance = abs(ymax - CoorXEntranceLine2)

    if AbsDistance <= 9:
        return 1
    else:
        return 0

api = '6139415220:AAHESYxYrE5O0-I9QXYtd08RuxxjgdoOGg' #token dari
bot telegram
bot = telepot.Bot(api)

LABELS = ['person','Aan','Mamah', 'Edo']

if __name__ == '__main__':

    parser = argparse.ArgumentParser()
    parser.add_argument("--model",           default="ssd.tflite",
help="Path of the detection model.")
    parser.add_argument("--num_threads",    type=int,    default=9,
help="Threads.")
    args = parser.parse_args()

    model        = args.model
    num_threads  = args.num_threads

    interpreter = Interpreter(model_path=model)
    interpreter.set_num_threads(num_threads)
```

```
    interpreter.allocate_tensors()
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()

    cam = cv2.VideoCapture(0)

    window_name = "Movie"
    cv2.namedWindow(window_name, cv2.WINDOW_AUTOSIZE)


    while True:
        start_time = time.perf_counter()

        ret, image = cam.read()

        # Resize and normalize image for network input
        image_height = image.shape[0]
        image_width = image.shape[1]
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame = np.expand_dims(frame, axis=0)
        frame = frame.astype(np.float32)
        cv2.normalize(frame, frame, -1, 1, cv2.NORM_MINMAX)
        cv2.putText(image,   fps,   (image_width   -   170,   15),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (38, 0, 255), 1, cv2.LINE_AA)
        cv2.putText(image,  detectfps,  (image_width  -  170,  30),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (38, 0, 255), 1, cv2.LINE_AA)

        # get results
        boxes                                                    =
interpreter.get_tensor(output_details[0]['index'])[0]
        classes                                                  =
interpreter.get_tensor(output_details[1]['index'])[0]
        scores                                                   =
interpreter.get_tensor(output_details[2]['index'])[0]
        count                                                    =
interpreter.get_tensor(output_details[3]['index'])[0]

        CoorXEntranceLine = int(image_width)
        cv2.line(image,     (int(CoorXEntranceLine),     0     ),
(int(CoorXEntranceLine), image_height), (255, 0, 100), 5)

        # draw boxes
        for  i,  (box,  classidx,  score)  in  enumerate(zip(boxes,
classes, scores)):
            probability = score

            if probability >= 0.45:
                if not box[0] or not box[1] or not box[2] or not
box[3]:
                    continue
                ymin = int(box[0] * image_height)
                xmin = int(box[1] * image_width)
                ymax = int(box[2] * image_height)
                xmax = int(box[3] * image_width)

                classnum = int(classidx)
                #print('coordinates:  ({},  {})-({},  {}).  class:
"{}". probability: {:.2f}'.format(xmin, ymin, xmax, ymax, classnum,
```

```
score))
                cv2.rectangle(image, (xmin, ymin), (xmax, ymax),
(0, 255, 0), 2)
                cv2.putText(image,                              '{}:
{:.2f}'.format(LABELS[classnum],  score),  (xmin,  ymin  -  5),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 0), 2)
                #ini tambahan
                cv2.imwrite('orang.jpg', image)

                date  =  datetime.datetime.now().strftime("%Y-%m-
%d")
                jam = datetime.datetime.now().strftime("%H:%M:%S")

                CoordXCentroid = int((xmax))
                cv2.circle(image,  ObjectCentroid,  1,  (255  ,255
,255), 10)

                if(CheckEntranceLineCrossing(CoordXCentroid,
CoorXEntranceLine)):

                    EntranceCounter += 1
                    if classnum == 0:
                        print('Person')

bot.sendPhoto("6269595626",photo=open("orang.jpg","rb"))    #untuk
mengirim gambar ke telegram #gianti
                        bot.sendMessage("6269595626",    "Person")
#untuk memberikan notif ke telegram
                        print('Person' ,date, jam)

                    elif classnum == 1:
                        print('Aan',date, jam)

bot.sendPhoto("6269595626",photo=open("orang.jpg","rb"))
                        bot.sendMessage("6269595626", "Aan")

                    elif classnum == 2:
                        print('Mamah' ,date, jam)

bot.sendPhoto("6269595626",photo=open("orang.jpg","rb"))
                        bot.sendMessage("6269595626", "Mamah")

                    elif classnum == 3:
                        print('Edo',date, jam)

bot.sendPhoto("6269595626",photo=open("orang.jpg","rb"))
                        bot.sendMessage("6269595626", "Edo")

                    else:
                        print('Tidak terdeteksi')

                if(CheckEntranceLineCrossing2(CoordXCentroid,
CoorXEntranceLine2)):

                    EntranceCounter2 += 1
                    if classnum == 0:
                        print('Person')

bot.sendPhoto("6269595626",photo=open("orang.jpg","rb"))    #untuk
```

```
mengirim gambar ke telegram #gianti
                        bot.sendMessage("6269595626",    "Person")
#untuk memberikan notif ke telegram
                        print('Person' ,date, jam)

                    elif classnum == 1:
                        print('Aan',date, jam)

bot.sendPhoto("6269595690",photo=open("orang.jpg","rb"))
                        bot.sendMessage("6269595626", "Aan")

                    elif classnum == 2:
                        print('Mamah' ,date, jam)

bot.sendPhoto("6269595690",photo=open("orang.jpg","rb"))
                        bot.sendMessage("6269595626", "Mamah")

                    elif classnum == 3:
                        print('Edo',date, jam)

bot.sendPhoto("6269595690",photo=open("orang.jpg","rb"))
                        bot.sendMessage("6269595626", "Edo")

                    else:
                        print('Tidak terdeteksi')

        cow = cv2.resize(image, (720,480))
        cv2.imshow(window_name, cow)


        if cv2.waitKey(1)&0xFF == ord('q'):
            break

        detectframecount += 1
        # FPS calculation
        framecount += 1
        if framecount >= 10:
            fps = "(Playback) {:.1f} FPS".format(time1 / 10)
            detectfps         =         "(Detection)         {:.1f}
FPS".format(detectframecount / time2)
            framecount = 0
            detectframecount = 0
            time1 = 0
            time2 = 0
        end_time = time.perf_counter()
        elapsedTime = end_time - start_time
        time1 += 1 / elapsedTime
        time2 += elapsedTime
```