Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.938446
Vol. 11 No. 3, Juni 2024, hlm. 547-556 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

DETEKSI OBJEK MANUSIA PADA CITRA MENGGUNAKAN SINGLE SHOT DETECTOR (SSD) BERBASIS EDGE COMPUTING

Muhammad Iqbal[*1], Dwi Marisa Midyanti[2], Syamsul Bahri[3]

[1,2,3]Universitas Tanjungpura, Pontianak
Email: [1]iniiqbal218@student.untan.ac.id, [2]dwi.marisa@siskom.untan.ac.id, [3]syamsul.bahri@siskom.untan.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 04 Januari 2024, diterima untuk diterbitkan: 25 April 2024)

Abstrak

Keamanan merupakan salah satu aspek penting di kehidupan manusia. Kemajuan teknologi yang ada dapat dimanfaatkan untuk meningkatkan keamanan, khususnya untuk kasus pencurian. Sistem kamera pengawas seperti CCTV telah terbukti dalam meningkatkan keamanan. Tetapi, CCTV mengharuskan pengawas untuk memantau layar CCTV secara manual 24/7 untuk melihat adanya pergerakan manusia. Pada penelitian ini, dibangun sistem deteksi secara otomatis menggunakan sensor *passive infrared* (PIR) HC-SR501 yang dikontrol oleh ESP32-CAM. Algoritma *single shot detector* (SSD) digunakan untuk mendeteksi objek pada citra dan *edge computing* digunakan untuk mengurangi *latency* pada proses transimisi data sehingga dapat memberikan informasi secara *real-time*. Proses transmisi data citra menggunakan protokol MQTT. Setiap ESP32-CAM akan menjadi *publisher* dan *edge* akan menjadi *subscriber*. Data citra yang digunakan berformat *jpg* dan memiliki resolusi 800x600 *pixels*. Sebanyak 1050 data digunakan untuk membangun model algoritma SSD. Seluruh data dibagi menjadi 3 bagian yaitu 735 data latih, 210 data evaluasi, dan 105 data uji. Data uji pada penelitian ini terdiri dari 3 jenis data, di antaranya 35 data uji siang hari, 35 data uji sore hari, dan 35 data uji malam hari. Algoritma SSD pada penelitian ini menghasilkan ketepatan deteksi 87.51% mAP pada data uji siang hari, 81.39% mAP pada data uji sore hari, dan 76.82% mAP pada data uji malam hari. Proses dari saat sensor HC-SR501 mendeteksi gerakan hingga informasi sampai ke user membutuhkan rata-rata waktu 2,843 detik.

Kata kunci: *edge computing, Single Shot Detector, ESP32-CAM, keamanan, MQTT*

HUMAN OBJECT DETECTION IN IMAGES USING SINGLE SHOT DETECTOR (SSD) BASED ON EDGE COMPUTING

Abstract

Security is one of the most important aspects of human life. Technological advances can be utilized to improve security, especially in the case of theft. Surveillance camera systems such as CCTV have been proven to improve security. However, CCTV requires the person to monitor the CCTV screen manually 24/7 to see any human movement. In this research, an automatic detection system was built using HC-SR501 passive infrared (PIR) sensor controlled by ESP32-CAM. The single shot detector (SSD) algorithm is used to detect an object on the image and edge computing is used to reduce the latency of the transmission process so that it can achieve real-time information. The transmission process of image data uses the MQTT protocol. Each ESP32-CAM will be the publisher and the edge will be the subscriber. The image data used are in jpg format and have a resolution of 800x600 pixels. A total of 1050 data were used to build the SSD algorithm model. All data is divided into 3 parts, namely 735 training data, 210 evaluation data, and 105 test data. The test data consists of 3 types of images, including 35 daytime images, 35 afternoon images, and 35 nighttime images. The SSD algorithm in this research resulted in 87.51% mAP on daytime images, 81.39% mAP on afternoon images, and 76.82% mAP on nighttime images. The process from the moment HC-SR501 sensor detects movement until the information reaches the user takes an average of 2.843 seconds.

Keywords: *edge computing, Single Shot Detector, ESP32-CAM, security. MQTT*

1. PENDAHULUAN

Keamanan merupakan salah satu hal yang sangat penting dalam kehidupan manusia. Dalam kehidupan sehari-hari, keamanan dapat berarti perlindungan dari bahaya fisik dan non-fisik seperti kecelakan, kebakaran, atau bahkan pencurian. Di era

yang semakin maju dan berkembang seperti sekarang, teknologi menjadi salah satu faktor yang dapat membantu meningkatkan keamanan (Albar, Ambarita and Ibrahim, 2019).

Salah satu teknologi yang dapat dimanfaatkan untuk membantu meningkatkan keamanan khususnya dalam kasus pencurian adalah CCTV kamera. CCTV kamera dapat merekam pergerakan atau aktivitas yang terjadi di suatu area namun tidak dapat membedakan antara pergerakan dari manusia dan bukan manusia. CCTV kamera biasanya tidak menyediakan *real-time* notifikasi, maka dari itu dibutuhkan pengawasan secara manual 24/7 (Pandya et al., 2018). Semakin banyak Layar CCTV yang diawasi, konsentrasi dari pengawas juga semakin menurun dan menyebabkan hampir tidak mungkin untuk memperhatikan keseluruhan layar CCTV untuk setiap waktu (Bhatti et al., 2021).

Alternatifnya, keamanan dapat ditingkatkan dengan memanfaatkan sistem deteksi gerakan menggunakan sensor PIR (*Passive Infra-Red*) dan kamera. Sensor PIR bekerja dengan mendeteksi perubahan suhu di sekitarnya, sehingga dapat dimanfaatkan untuk mendeteksi gerakan manusia atau hewan (Toyib et al., 2019). Kemudian, kamera dapat merekam citra dari area yang diamati dan pemrosesan citra dapat digunakan untuk menganalisis citra dan mendeteksi objek yang ada di dalamnya.

Kamera pada umumnya tidak dilengkapi dengan fitur objek deteksi. Oleh karena itu, untuk mendeteksi adanya objek pada citra yang dihasilkan oleh kamera, diperlukan algoritma pemrosesan citra. Salah satu algoritma tersebut adalah algoritma *deep learning* yang memiliki performa lebih baik dalam mendeteksi objek dibandingkan dengan algoritma pendahulunya (Wu, Sahoo and Hoi, 2020).

Dalam implementasinya, teknologi deteksi gerakan dan deteksi objek yang menggunakan algoritma *deep learning* ini membutuhkan daya pemrosesan yang besar untuk dapat melakukan tugasnya dengan efektif (Marchisio et al., 2019). Salah satu cara untuk mengoptimalkan perangkat keras dalam pendeteksian objek adalah dengan menggunakan arsitektur *edge computing.* Arsitektur ini memungkinkan pengolahan data yang dilakukan lebih dekat dengan sumber data. Penggunaan arsitektur ini agar pemrosesan data dapat dilakukan secara *real-time* dan memiliki latensi rendah dalam pengiriman data (Zhou et al., 2019).

Penelitian tentang deteksi gerakan dalam meningkatkan aspek keamanan telah dilakukan. Salah satunya adalah penelitian yang dilakukan oleh (Albar, Ambarita and Ibrahim, 2019). Penelitian tersebut membangun sistem deteksi gerakan menggunakan sensor PIR HC-SR501. Sensor PIR diletakkan di atas pintu menghadap ke bawah untuk melakukan pendeteksian gerakan dan berhasil mendeteksi gerakan dari jarak 1 meter hingga 5 meter.

Penelitian lain yang menggunakan Model *deep learning* seperti *VGG16, InceptionV3, Inception ResnetV2, SSD MobileNetV1, YoloV3, Faster RCNN-Inception Resnet V2* dan *YoloV4* untuk mendeteksi objek pada citra juga telah dilakukan. Penelitian ini bertujuan untuk mendeteksi adanya objek senjata api atau tidak pada citra. Penelitian ini berhasil mencapai performa terbaik menggunakan model *YoloV4* dengan 91.73% mAP (Bhatti et al., 2021).

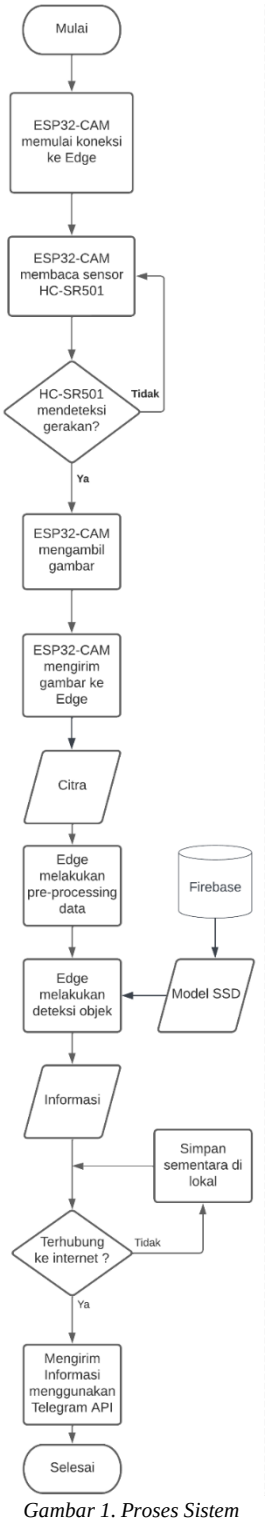
Penelitian lain terkait algoritma deep learning dilakukan oleh Cruz dkk. Penelitian ini menggunakan Model deep learning YoloV5 untuk melakukan object recognition pada tanaman strawberry. Penelitian Cruz juga menerapkan arsitektur edge computing dan menjalankan seluruh proses pengolahan citra pada edge. Proses transmisi data nya menggunakan modul LoRa. Penelitian ini berhasil mencapai waktu rata-rata dari proses pada edge 2 detik dan ketepatan deteksi pada model YoloV5 sebesar 0.975% mAp (Cruz et al., 2022).

Penelitian lain yang menggabungkan algoritma deep learning, kamera dan sensor PIR dilakukan oleh Alif dkk. Penelitian ini menggunakan model deep learning EfficientNetB4 untuk melakukan face recognition sebagai upaya dalam mendeteksi penyusup. Sistem yang dibangun menggunakan satu Raspberry Pi untuk satu kamera dalam melakukan face recognition-nya. Penelitian ini berhasil mencapai accuracy sebesar 97.6%. (Ali, Naif and Humood, 2023).

Berdasarkan latar masalah yang dipaparkan dan penelitian-penelitian sebelumnya, penelitian yang akan dilakukan adalah memanfaatkan arsitektur *edge computing* untuk menjalankan algoritma *deep learning*. Model *deep learning* yang digunakan adalah SSD *MobileNetV2*. Penelitian ini memanfaatkan sensor PIR dan kamera dengan bantuan algoritma *deep learning* untuk melakukan deteksi awal pada objek manusia. Penerapan edge computing bertujuan agar tercapainya deteksi manusia secara real-time pada beberapa area dan informasi tidak akan hilang karena sistem akan menyimpan informasi secara lokal saat terjadi gangguan internet. Hal ini dapat membantu otomatisasi pengawasan dari beberapa area dengan memberikan informasi kondisi lingkungan pada area tersebut saat terdeteksinya manusia.

2. METODE PENELITIAN

Pada penelitian ini, sensor PIR dimanfaatkan sebagai otomatisasi sistem dan kamera pada ESP32-CAM dimanfaatkan sebagai sumber data citra. Data citra akan diolah dan dideteksi pada *edge* menggunakan model SSD *MobileNetV2*. Gambar 1 menunjukan alur proses sistem pada penelitian ini.



Gambar 1. Proses Sistem

2.1. *Single Shot Detector & MobileNetV2*

Single shot detector (SSD) adalah algoritma dan arsitektur yang menggunakan *deep convolutional neural network* (CNN) sebagai basisnya. Algoritma ini dirancang khusus untuk tugas *object detection.* SSD berkerja dengan cara melakukan *feed-forward convolutional network* yang akan menghasilkan kotak-kotak pembatas dan skor untuk objek yang terdeteksi/terklasifikasi di kotak tersebut. Setelah itu, SSD akan menerapkan *non-maximum supression* untuk menghasilkan deteksi akhir (Liu et al., 2015).

MobileNetV2 merupakan arsitektur yang dibangun menggunakan CNN. Arsitektur ini memiliki total parameter bobot yang lebih sedikit daripada arsitektur-arsitektur pendahulunya seperti *MobileNetV1* dan *VGG16* namun tetap menghasilkan tingkat akurasi yang cukup baik (Sandler et al., 2018). *MobileNetV2* digunakan pada SSD sebagai *backbone network* yang berfungsi untuk mengekstraksi fitur dari citra.

2.2. Protokol MQTT

MQTT adalah sebuah protokol komunikasi *Pub*/*Sub* berbasis TCP/IP yang simpel dan ringan. MQTT dapat melakukan komunikasi *one-to-many*, *two-way* bahkan komunikasi asinkronus (Mishra, Mishra and Kertesz, 2021). Protokol ini memiliki 3 komponen utama, yaitu:

1. *Publisher*, berperan sebagai klien MQTT yang mengirim data.
2. *Subscriber*, berperan sebagai klien MQTT yang menerima data.
3. *Broker*, berperan sebagai perantara antara *publisher dan subscriber*

2.3. Telegram API

Telegram adalah aplikasi layanan pengiriman dan penerimaan pesan. Telegram memiliki layanan API yang dapat diakses oleh para *developers* untuk membangun aplikasi yang terhubung dengan platform Telegram. Layanan ini sering digunakan oleh para *developers* untuk membangun sistem manajemen pesan dan chatbot (Nugraha and Sebastian, 2021). Telegram API digunakan pada penelitian ini untuk mengirim informasi hasil deteksi melalui *bot.*

2.4. Analisis Kebutuhan

Analisi kebutuhan dilakukan untuk menentukan kebutuhan yang diperlukan dalam melaksanakan penelitian dan membangun sistem. Adapun kebutuhan perangkat keras terdiri dari ESP32-CAM, Sensor PIR HC-SR501, dan Komputer dengan spesifikasi CPU I7-6700HQ 2.80ghz, GPU Nvidia GTX 1060 3GB, RAM 8GB ddr4, dan dengan fitur WI-FI. Adapun kebutuhan perangkat lunak terdiri dari bahasa pemrograman Python, Arduino IDE, Tensorflow, dan MQTT mosquitto *broker.*

2.5. Pengumpulan data

Penelitian ini menggunakan sumber data primer. Data citra diambil menggunakan perangkat keras yang sama dengan yang digunakan pada penelitian ini, yaitu data citra dari ESP32-CAM. Data citra diambil dengan sudut kemiringan ESP32-CAM $10° - 30°$ menghadap ke bawah. Data yang dikumpulkan berjumlah 1050 total data. Data ini

dibagi menjadi 3 bagian, diantaranya, 735 data (70%) untuk data latih, 210 data (20%) untuk data evaluasi dan 105 data (10%) untuk data uji. Data citra yang dikumpulkan berformat *jpg* dan memiliki resolusi 800x600 piksel. Keseluruhan data ini diambil dari rentang waktu 09:00 sampai 21:00.

2.6. Metrik Evaluasi

2.6.1 *Delay*

Delay menunjukan waktu yang dibutuhkan oleh data saat dikirim hingga sampai ke tujuannya. *Delay* dihitung dengan cara mengurangkan waktu saat data sampai ke tujuan dengan waktu saat data mulai dikirim. Rata-rata dari *delay* dapat dihitung dengan merata-rata kan waktu saat data sampai ke tujuan dikurangkan dengan waktu saat data mulai dikirim (Khan et al., 2020). Rata-rata *delay* dijabarkan dengan persamaan 1.

$$D_{avg} = Tr_{avg} - Ts_{avg} \quad (1)$$

Keterangan:
D_{avg} : Rata-rata *delay*.
Tr_{avg} : Rata-rata waktu data saat sampai ke tujuan.
Ts_{avg} : Rata-rata waktu data saat mulai dikirim.

2.6.2 *Confusion Matrix*

Confusion matrix adalah sebuah metode perhitungan untuk mengevaluasi performa dari sistem. *Confusion matrix* membandingkan nilai hasil klasifikasi yang dilakukan oleh sistem dengan nilai sebenarnya. *Confusion matrix* adalah matriks dua dimensi yang biasanya digunakan pada dua variabel kelas, satu untuk kelas positif, satu untuk kelas negatif (Düntsch and Gediga, 2019). Nilai yang dihasilkan oleh *Confusion matrix* dapat digunakan untuk menghitung *precision* dan *recall*. Penggunaan rumus *confusion matrix* dijabarkan pada Tabel 1

Tabel 1. *Confusion Matrix*

Class	***Positive***	***Negative***
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Perhitungan *precision* dijabarkan dengan persamaan 2 dan *recall* dijabarkan dengan persamaan 3.

$$\text{Precision} = \frac{TP}{TP + FP} \cdot 100\% \quad (2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \cdot 100\% \quad (3)$$

Keterangan:
TP : *True Positive*, jumlah nilai prediksi positif yang sesuai dengan nilai sebenarnya.
FP : *False Positive*, jumlah nilai prediksi positif yang tidak sesuai dengan nilai sebenarnya.
TN : *True Negative*, jumlah nilai prediksi negatif yang sesuai dengan nilai sebenarnya.
FN : *False Negative*, jumlah nilai prediksi negatif yang tidak sesuai dengan nilai sebenarnya.

2.6.3 *Mean Average Precision* (mAP)

mAP adalah salah satu metrik yang sering digunakan untuk mengevaluasi model *deep learning*. Metrik ini menghitung rata-rata area dari kurva *Precission-Recall* untuk tiap label atau yang biasanya disebut *average precision* (AP) (Padilla, Netto and da Silva, 2020). Berdasarkan (Wang, 2022) berikut adalah langkah untuk menghitung mAP:

1. Pasangkan *bounding box* hasil deteksi dengan *ground truth box* berdasarkan nilai IoU tertinggi nya.
2. Urutkan setiap bounding box berdasarkan *confidence score* nya secara *descending* dan Tetapkan *bounding box* sebagai *true positive* jika label dari *bounding box* tersebut sama dengan label pada *ground truth box* dan nilai IoU nya di atas 0.5. Tetapkan sebagai *false positive* pada *bounding box* lain yang berpasangan pada *ground truth box* yang sama dan nilai IoU nya di bawah 0.5.
3. Hitung *precision* dan *recall* dari *bounding box true positive*, *bounding box false positive*, dan *bounding box false negative* dengan *confidence score* tertinggi hingga terkecil secara kumulatif.
4. Hitung *average precision* (AP) dengan menggunakan persamaan 4.

$$\text{AP} = \sum_{i=0}^{n} max_{\tilde{\imath} \geq i} P(\tilde{\imath}) \Delta r(i) \quad (4)$$

Keterangan:
n : Jumlah data.
$max_{\tilde{\imath} \geq i} P(\tilde{\imath})$: Nilai *precision* tertinggi dari list data $\tilde{\imath}$ yang lebih besar atau sama dengan i.
$\Delta r(i)$: Perubahan nilai *recall* untuk data ke i.

5. Hitung *mean average precision* (mAP) dengan menggunakan persamaan 5.

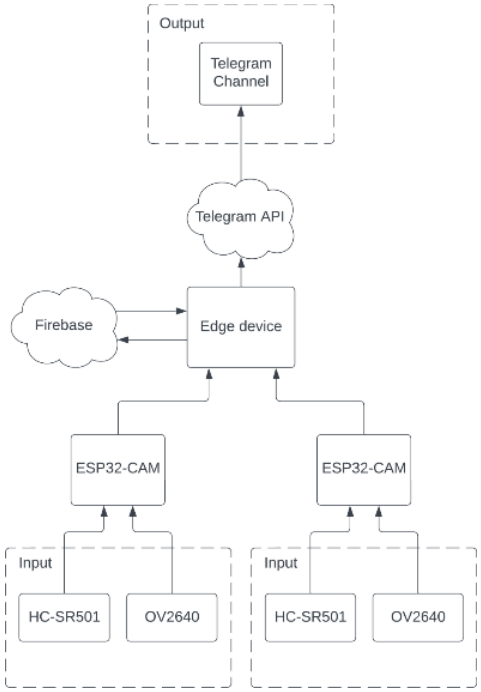
$$\text{mAP} = \frac{1}{c} \sum_{0}^{c} AP_C \quad (5)$$

Keterangan:
c : Jumlah label.
AP_C : Nilai *average precision* untuk label c

3. PERANCANGAN DAN IMPLEMENTASI

Perancangan arsitektur sistem pada penelitian ini terdiri dari perancangan perangkat keras dan perangkat lunak. Perancangan pada perangkat keras meliputi perancangan dari sistem kamera dan sensor pada EPS32-CAM, sedangkan perancangan

perangkat lunak meliputi perancangan dari perangkat lunak yang akan dijalankan pada ESP32-CAM, perancangan model SSD untuk objek deteksi dan perancangan perangkat lunak yang ada pada *edge*. Arsitektur Sistem ini dapat dilihat pada Gambar 2.



Gambar 2. Arsitektur Sistem

3.1. Perancangan Perangkat Keras

Perancangan perangkat keras untuk sensor PIR dan kamera dilakukan dengan menghubungkan sensor HC-SR501 dengan ESP32-CAM menggunakan *external pins*. Konfigurasi *pin* antar HC-SR501 dan ESP32-CAM dapat dilihat pada Tabel 2.

Tabel 2. Konfigurasi *Pin* HC-SR501 pada ESP32-CAM

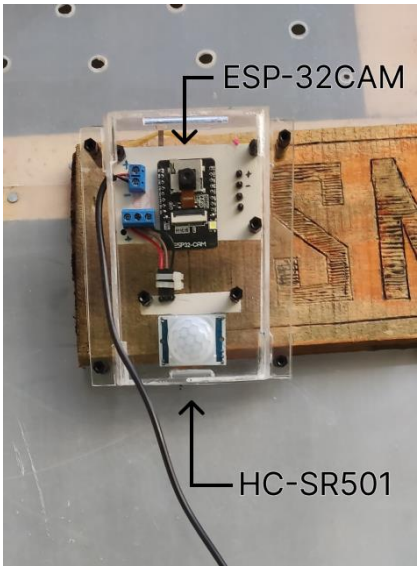
HC-SR501	ESP32-CAM
GND	GND
OUT	GPIO13
VCC	5V

Setelah itu, menghubungkan modul kamera OV2640 dengan ESP32-CAM menggunakan inner pins berdasarkan skematik yang ada pada ESP32-CAM. Konfigurasi pin antar OV2640 dan ESP32-CAM dapat dilihat pada Tabel 3.

Tabel 3. Konfigurasi *Pin* OV2640 pada ESP32-CAM

OV2640	ESP32-CAM
DO	GPIO 5
D1	GPIO 18
D2	GPIO 19
D3	GPIO 21
D4	GPIO 36
D5	GPIO 39
D6	GPIO 34
D7	GPIO 35
XCLK	GPIO 0
PCLK	GPIO 22
VSYNC	GPIO 25
HREF	GPIO 23
SDA	GPIO 26
SCL	GPIO 27
POWERPIN	GPIO 32

Implementasi perangkat keras dilakukan dengan menghubungkan komponen-komponen yang telah dirancang. Sensor HC-SR501 yang diletakkan tepat di bawah ESP32-CAM berfungsi sebagai pendeteksi pertama dari gerakan yang memicu ESP32-CAM untuk menangkap dan mulai mengirim citra ke edge. Posisi HC-SR501 mengarah pada sisi dan sudut yang sama dengan modul Kamera ESP32-CAM agar gambar yang ditangkap sesuai dengan lingkungan yang memicu aktifnya sensor. Implementasi Perangkat keras dapat dilihat pada Gambar 3.



Gambar 3. Implementasi Perangkat Keras

3.2. Perancangan Perangkat Lunak

Perancangan perangkat lunak pada penelitian ini dibuat agar sistem dapat berfungsi sesuai dengan tujuan penelitian. Pada ESP32-CAM, perangkat lunak dirancang agar ESP32-CAM dapat menangkap data citra saat sinyal keluaran yang dikirim oleh sensor HC-SR501 tinggi (*HIGH*). Gambar yang ditangkap akan dikirim ke *edge* yang terhubung di jaringan yang sama. Pada *edge*, terdapat dua perangkat lunak yang dibangun. Perangkat lunak pertama untuk menjalankan model SSD. Model ini akan dilatih terlebih dahulu secara terpisah hingga memiliki bobot yang terbaik untuk mendeteksi objek. Perangkat lunak kedua adalah program untuk menerima data citra yang dikirim dari ESP32-CAM, melakukan *post-processing* pada data citra, dan melakukan deteksi pada data citra menggunakan model SSD.

Implementasi perangkat lunak dilakukan dengan membangun model *deep learning* SSD, memprogram ESP32-CAM dan *Edge* agar sesuai

dengan tujuan penelitian. Awalnya, ESP32-CAM diprogram untuk mengumpulkan data citra yang diperlukan untuk melatih model. Data latih ini digunakan dalam pembangunan SSD. Model dengan performa terbaik disimpan menggunakan *storage* dari firebase

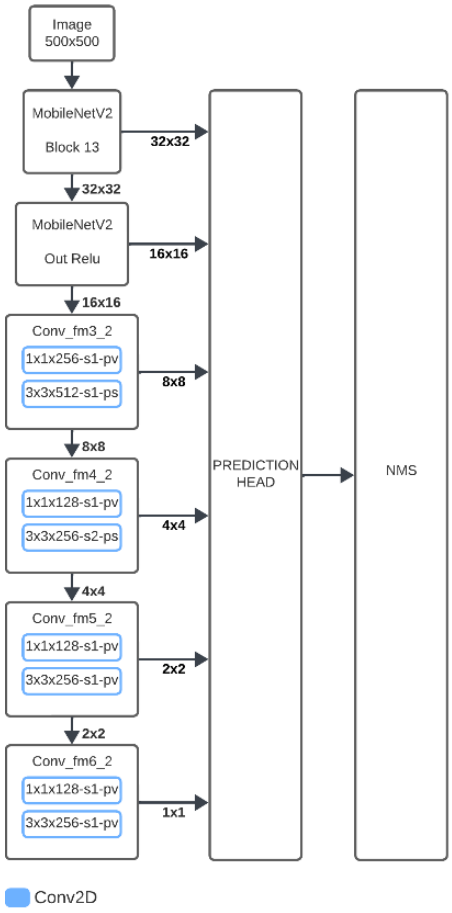
Tahap selanjutnya, ESP32-CAM diprogram ulang dengan berperan sebagai MQTT *publisher*. Terdapat 3 topik yang ditetapkan pada setiap ESP32-CAM, 3 topik ini berfungsi untuk mengirim data citra dan data waktu. Saat HC-SR501 mendeteksi perubahan sinar inframerah, ESP32-CAM akan menangkap dan mengirim data citra ke *edge*.

Edge berperan sebagai MQTT *subscriber*. Setiap *publisher* yang mengirim data terhadap topik terkait akan diterima oleh *edge*. Data citra yang diterima akan di-*preprocessing* terlebih dahulu, setelah itu, *edge* akan mendeteksi citra tersebut menggunakan model SSD yang telah diunduh dari firebase. Saat SSD mendeteksi adanya objek manusia, *edge* akan mengirim citra tersebut ke *channel* Telegram sebagai *bot*. Tampilan dari *bot* Telegram dapat dilihat pada Gambar 4.

Gambar 4. Implementasi Perangkat Keras

3.3. Perancangan *Model* SSD *MobileNetV2*

Feature extractor dari SSD dibangun menggunakan sebagian dari arsitektur MobileNetV2 dan dengan tambahan 8 *layer* konvolusi yang terbagi menjadi 4 bagian. Setiap bagian dari *feature extractor* berfungsi untuk menghasilkan *output feature map* dari *input* data citra. *Output* dari tiap bagian pada *feature extractor* akan dihubungkan dengan layer konvolusi pada bagian *prediction head*. *Layer* konvolusi yang ada pada *prediction head* berfungsi untuk menghasilkan nilai dari deteksi label objek dan lokasi objek. *Layer* NMS akan menghasilkan hasil akhir dari deteksi dan yang akan digunakan sebagai informasi terdeteksi nya objek. Gambar 5 menunjukan arsitektur SSD yang digunakan pada penelitian ini.

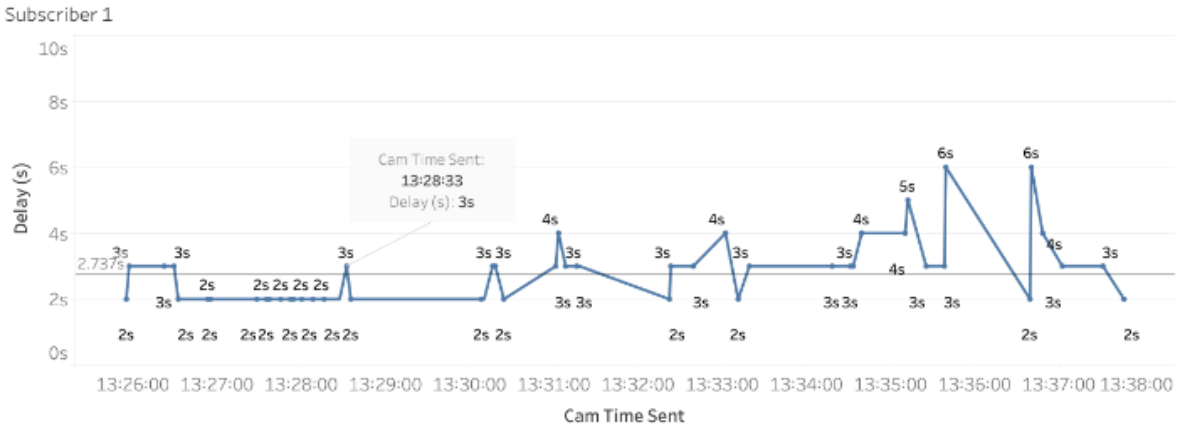


Gambar 5. Arsitektur SSD MobileNetV2

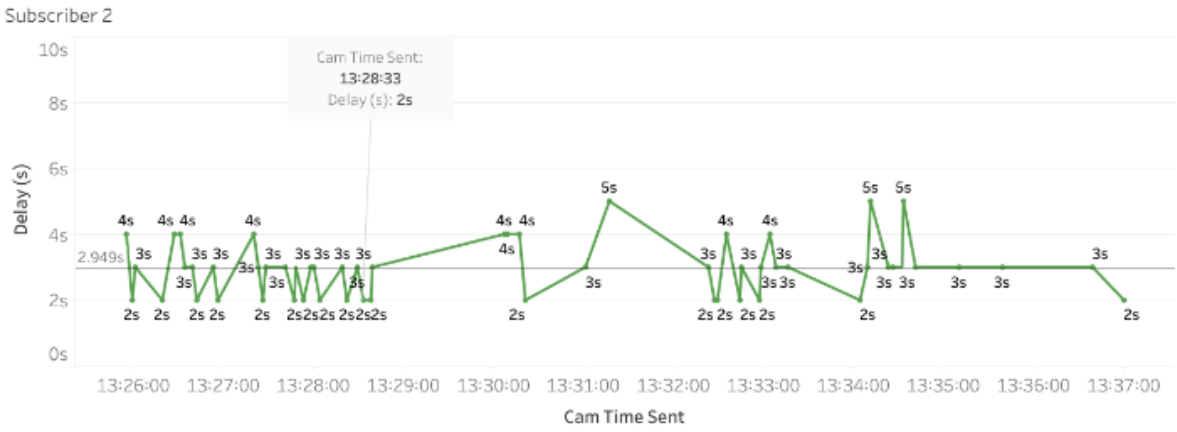
Anchors atau *default bounding box* digunakan pada algoritma SSD sebagai referensi dari *ground truth box*. Dimensi *anchors* dibuat sedemikian rupa menyerupai bentuk dimensi dari setiap deteksi yang dihasilkan pada bagian *prediction head*. *Anchors* dibuat dengan beragam bentuk dan ukuran sesuai dengan *scales* dan *aspect ratio* yang ditetapkan. Pada penelitian ini, terdapat 4 sampai 6 bentuk atau ukuran dari *anchors* dengan nilai dari *aspect ratio* yang digunakan adalah $1, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}$. Penetapan nilai *aspect ratio* tersebut bertujuan agar *anchors* yang dibuat memiliki bentuk atau area yang mirip dengan objek manusia pada data citra.

4. HASIL DAN PEMBAHASAN

Proses deteksi objek manusia pada citra pada penelitian ini dilakukan dengan menggunakan algoritma SSD serta dengan memanfaatkan ESP32-CAM dan sensor HC-SR501 sebagai sumber data citra nya. Hasil yang dibahas dalam bab ini merupakan hasil pengujian dari berbagai metrik evaluasi.



Gambar 6. *Delay* Seluruh Proses Pada *Subscriber* 1



Gambar 7. *Delay* Seluruh Proses Pada *Subscriber* 2

Pengujian *delay* dilakukan untuk mengetahui waktu yang dibutuhkan oleh sistem dalam mengirim dan mengolah data hingga sampai ke sisi *user* melalui Telegram API. Tabel 4 adalah rata-rata nilai *delay* untuk setiap jenis proses yang terjadi pada penelitian ini.

Tabel 4. Rata-rata *Delay* Setiap Proses

	***Size* (byte)**	***Delay* ToEdge (s)**	***Inference* (s)**	***Delay* ToTel (s)**
Subscriber 1	34785	0.842	0.421	2.737
Subscriber 2	41843	0.780	0.339	2.949

Pada tabel 4, *DelayToEdge* adalah rata-rata waktu yang dibutuhkan oleh ESP32-CAM untuk mengirim data citra hingga sampai ke *edge*, *Inference* adalah rata-rata waktu *edge* melakukan deteksi pada data citra, dan *DelayToTel* adalah rata-rata waktu yang dibutuhkan oleh sistem dari terdeteksinya gerakan hingga informasi sampai ke Telegram.

Protokol MQTT memberikan kemudahan dalam penerapan arsitektur edge computing. Konsep *subscribe-publish* pada topik yang telah ditentukan sesuai dengan konsep *Edge Computing*. Beberapa *edge* akan menjadi *broker* dan setiap *edge* akan menjadi *subscriber*. *Edge* akan menerima data yang dikirim oleh ESP32-CAM. Satu *edge device* bisa menjadi lebih dari satu *subscriber* dengan cara menjalankan program *subscribe* yang sama dengan ID dan topik yang berbeda. Proses transmisi data dari *publisher* terhadap *subscriber* pada penelitian ini membutuhkan waktu rata-rata 0,842 detik untuk *subcriber* 1 dan 0,78 detik untuk *subscriber* 2 dengan ukuran data rata-rata sebesar 34785 *byte* pada *subscriber* 1 dan 41843 *byte* pada subscriber 2. Gambar 6 dan 7 menunjukan hasil dari pengujian *delay* seluruh proses pada sistem.

Pengujian *confusion matrix* bertujuan untuk mengetahui seberapa tepat model SSD dalam melakukan deteksi nya. Pada kasus objek deteksi, penetapan *true positive* atau *false positive* dari suatu data dihitung menggunakan nilai IoU nya pada *ground truth box* yang ada. Tabel 5 adalah nilai dari hasil *confusion matrix*.

Tabel 5. Nilai *Confusion Matrix*

Waktu	**TP**	**FP**	**FN**	**TN**
Siang	50	4	7	N/A
Sore	49	2	11	N/A
Malam	48	6	13	N/A

Pada Tabel 5, nilai dari *true negative* (TN) tidak digunakan karena tidak relevan untuk kasus objek deteksi. Kasus objek deteksi berfokus pada

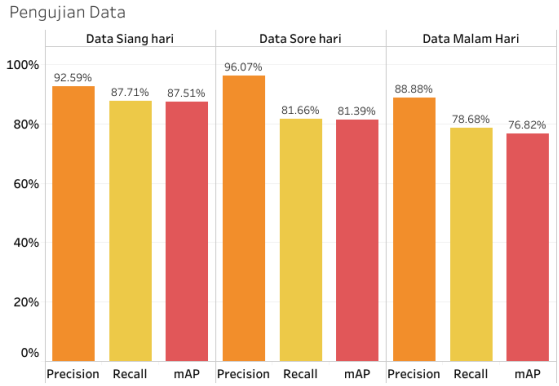
mendeteksi adanya objek bukan ketiadaan objek. Maka dari itu TN dituliskan sebagai *Not Applicable* (N/A) pada seluruh data uji.

Pengujian dan pengukuran jarak deteksi oleh sistem bertujuan untuk mengetahui batasan dari kinerja sistem. Berdasarkan pengujian yang dilakukan, didapati jarak pergerakan yang terdeteksi oleh HC-SR501 dan jarak deteksi objek manusia yang dihasilkan oleh SSD MobileNet V2 pada citra yang ambil oleh ESP32-CAM ditunjukan pada Tabel 6.

Tabel 6. Jarak Deteksi

Jarak (meter)	HC-SR501	SSD *MobileNetV2*
1	ya	ya
2	ya	ya
3	ya	ya
4	ya	ya
5	ya	ya
6	ya	ya
7	ya	ya
8	tidak	ya
9	tidak	tidak

Pengujian mAP bertujuan untuk mengevaluasi model SSD yang dibangun. Metrik mAP tidak hanya memberikan informasi mengenai seberapa tepat model SSD yang dibangun tetapi juga memberikan informasi mengenai kemampuan model SSD dalam melakukan deteksi yang relevan. Gambar 8 menunjukan performa dari model SSD *MobileNetV2* pada penelitian ini.



Gambar 8. *Precision*, *Recall*, dan mAP MobileNetV2

Pada Gambar 8, terlihat adanya penurunan kemampuan deteksi oleh model pada saat malam hari. Penurunan ini disebabkan karena terbatasnya cahaya pada area yang diawasi saat malam hari. Keterbatasan ini menyebabkan ESP32-CAM menghasilkan kualitas citra yang kurang baik dan mempengaruhi model dalam mengidentifikasi dan mendeteksi objek pada citra.

5. KESIMPULAN

Berdasarkan pengujian dan pembahasan yang dilakukan, model SSD MobileNetV2 pada penelitian ini menghasilkan ketepatan deteksi sebesar 87.51% mAP pada data siang hari, 81.66% mAP pada sore hari, dan 76.68% mAP pada malam hari. Implementasi arsitektur *edge computing* pada penelitian ini memiliki *delay* rata-rata sebesar 2,737 detik pada subscriber 1 dan 2,949 detik pada subscriber 2. Hal ini berarti *user* akan menerima informasi terdeteksi nya manusia pada area yang diamati kira-kira 3 detik lebih lama daripada waktu aktual nya.

Algoritma untuk object detection yang menggunakan deep learning utamanya bergantung pada kualitas data citra yang menjadi dataset dan yang akan dilakukannya object detection. Maka dari itu, peningkatan modul kamera atau memperbanyak variasi dataset dapat dipertimbangkan untuk penelitian selanjutnya. Saran ini diharapkan dapat meningkatkan performa dari algoritma untuk object detection yang akan dilakukan.

DAFTAR PUSTAKA

ALBAR, B., AMBARITA, A. AND IBRAHIM, A., 2019. Sistem Keamanan Ruangan Laboratorium Politeknik Sains dan Teknologi Wiratama Maluku Utara Menggunakan Sensor PIR (Passive Infra Red) dengan Metode Pengembangan Prototyping Berbasis Mikrokontroller ATmega328. *Jurnal Ilmiah ILKOMINFO - Ilmu Komputer & Informatika*, 2(2). https://doi.org/10.47324/ilkominfo.v2i2.34.

ALI, H.H., NAIF, J.R. AND HUMOOD, W.R., 2023. A New Smart Home Intruder Detection System Based on Deep Learning. *Al-Mustansiriyah Journal of Science*, 34(2), pp.60–69. https://doi.org/10.23851/mjs.v34i2.1267.

BHATTI, M.T., KHAN, M.G., ASLAM, M. AND FIAZ, M.J., 2021. Weapon Detection in Real-Time CCTV Videos Using Deep Learning. *IEEE Access*, 9, pp.34366–34382. https://doi.org/10.1109/ACCESS.2021.3059170.

CRUZ, M., MAFRA, S., TEIXEIRA, E. AND FIGUEIREDO, F., 2022. Smart Strawberry Farming Using Edge Computing and IoT. *Sensors*, 22(15), p.5866. https://doi.org/10.3390/s22155866.

DÜNTSCH, I. AND GEDIGA, G., 2019. Confusion Matrices and Rough Set Data Analysis. *Journal of Physics: Conference Series*, 1229(1), p.012055. https://doi.org/10.1088/1742-6596/1229/1/012055.

KHAN, I.U., QURESHI, I.M., AZIZ, M.A., CHEEMA, T.A. AND SHAH, S.B.H., 2020. Smart IoT Control-Based Nature Inspired Energy Efficient Routing Protocol for Flying Ad Hoc Network (FANET). *IEEE Access*, 8, pp.56371–56378. https://doi.org/10.1109/ACCESS.2020.2981531.

LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y. AND BERG, A.C., 2015. SSD: Single Shot MultiBox Detector. https://doi.org/10.1007/978-3-319-46448-0_2.

MARCHISIO, A., HANIF, M.A., KHALID, F., PLASTIRAS, G., KYRKOU, C., THEOCHARIDES, T. AND SHAFIQUE, M., 2019. Deep Learning for Edge Computing: Current Trends, Cross-Layer Optimizations, and Open Research Challenges. In: *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE. pp.553–559. https://doi.org/10.1109/ISVLSI.2019.00105.

MISHRA, B., MISHRA, B. AND KERTESZ, A., 2021. Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies*, 14(18), p.5817. https://doi.org/10.3390/en14185817.

NUGRAHA, K.A. AND SEBASTIAN, D., 2021. Designing Consultation Chatbot Using Telegram API and Webhook-based NodeJS Applications. In: *Proceedings of the 7th International Conference on Education and Technology (ICET 2021)*. [online] Atlantis Press. pp.119–122. https://doi.org/10.2991/assehr.k.211126.047.

PADILLA, R., NETTO, S.L. AND DA SILVA, E.A.B., 2020. A Survey on Performance Metrics for Object-Detection Algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE. pp.237–242. https://doi.org/10.1109/IWSSIP48289.2020.9145130.

PANDYA, S., GHAYVAT, H., KOTECHA, K., AWAIS, M., AKBARZADEH, S., GOPE, P., MUKHOPADHYAY, S. AND CHEN, W., 2018. Smart Home Anti-Theft System: A Novel Approach for Near Real-Time Monitoring and Smart Home Security for Wellness Protocol. *Applied System Innovation*, 1(4), p.42. https://doi.org/10.3390/asi1040042.

SANDLER, M., HOWARD, A., ZHU, M., ZHMOGINOV, A. AND CHEN, L.-C., 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks.

TOYIB, R., BUSTAMI, I., ABDULLAH, D. AND ONSARDI, O., 2019. Penggunaan Sensor Passive Infrared Receiver (PIR) Untuk Mendeteksi Gerak Berbasis Short Message Service Gateway. *Pseudocode*, 6(2), pp.114–124. https://doi.org/10.33369/pseudocode.6.2.114-124.

WANG, B., 2022. A Parallel Implementation of Computing Mean Average Precision.

WU, X., SAHOO, D. AND HOI, S.C.H., 2020. Recent advances in deep learning for object detection. *Neurocomputing*, 396, pp.39–64. https://doi.org/10.1016/j.neucom.2020.01.085.

ZHOU, Z., CHEN, X., LI, E., ZENG, L., LUO, K. AND ZHANG, J., 2019. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of the IEEE*, 107(8), pp.1738–1762. https://doi.org/10.1109/JPROC.2019.2918951.

Halaman ini sengaja dikosongkan.