Implementasi Algoritma SSD MobileDets untuk Mendeteksi Aktivitas Operator Memasuki Area Hazard di Sekitar Mesin

Tugas Akhir

Oleh:

Abdillah Fikri (4222001029)

Program Studi Teknik Robotika Jurusan Teknik Elektro Politeknik Negeri Batam

2023

Pernyataan Keaslian Tugas Akhir

Saya yang bertandatangan dibawah ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya yang berjudul : "Implementasi Algoritma SSD MobileDets untuk Mendeteksi Aktivitas Operator Memasuki Area Hazard di Sekitar Mesin" adalah **hasil karya sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan,** dan **bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.** Semua referensi yang dikutip atau dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan saya ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Batam, 12 Januari 2024

Abdillah Fikri
NIM: 4222001029

Lembar Pengesahan

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar
Sarjana Terapan Teknik (S.Tr.T)
di
Politeknik Negeri Batam

Oleh:
Abdillah Fikri (4222001029)

Dengan judul:
Implementasi Algoritma SSD MobileDets untuk Mendeteksi Aktivitas Operator Memasuki Area Hazard di Sekitar Mesin

Tanggal Sidang: 21 12, 2023

Disetujui oleh :

Dosen Penguji I

1. Ir. Ahmad Riyad Firdaus, Ph.D
NIK: 100013

Dosen Pembimbing

1. Eko Rudiawan Jamzuri, S.ST., M.Sc
NIK: 113117

Dosen Penguji II

2. Ryan Satria Wijaya, S.Tr.T., M.Tr.T.
NIK: 121249

Implementasi Algoritma SSD MobileDets untuk Mendeteksi Aktivitas Operator Memasuki Area Hazard di Sekitar Mesin

Abdillah Fikri, dan Eko Rudiawan Jamzuri*

Department of Electrical Engineering, Politeknik Negeri Batam
Jl. Ahmad Yani, Kel. Tlk. Tering, Kec. Batam Kota, Kota Batam, Kepulauan Riau, 29461, Indonesia
**Corresponding author. Email: ekorudiawan@polibatam.ac.id*

Abstract— Safety laser scanners are the main sensors proposed to protect work accidents in the industrial sector. However, they are expensive and limited to a 2D scanning area. Therefore, in this final project, research is carried out related to low-cost sensors using monocular cameras and Edge TPU as an alternative security laser scanner. The system can inform safety warnings using green, yellow, and red indicator lights, and use warning sounds using a buzzer in dangerous conditions. I captured the safe area using a monocular camera and processed the MobileDets SSD deep learning object detection image. To be able to detect objects, the person data that has been collected will be carried out a model training process, the model training process is carried out in order to produce a model that is ready for use. During the training process, the model obtained an mAP value of 0.217% on the validation data, and obtained an mAP value of 0.274% on the test data. Next, the object detector was run on Edge TPU, a USB device to accelerate the deep learning process. The resulting object detector model can detect objects in the form of people with an average accuracy of 73,81%. The object detector will recognize people and mark them with a bounding box. From the resulting bounding box, the object coordinates are then calculated to analyze people entering the safe area. When a person is detected entering a safe area, the system will provide a warning with the resulting indicator light. In addition, this system can generate security signals using USB Input/Output that can be integrated with machine or robot controllers.

Keywords— work safety; convolutional neural network; deep learning; Edge TPU.

Abstrak— Pemindai laser keselamatan adalah sensor utama yang diusulkan untuk melindungi kecelakaan kerja di sektor industri. Namun, sensor ini mahal dan terbatas di area pemindaian 2D. Oleh karena itu, dalam tugas akhir ini, dilakukan penelitian terkait sensor berbiaya rendah menggunakan kamera bermata dan *Edge TPU* sebagai pemindai laser keamanan alternatif. Sistem ini dapat menginformasikan peringatan keselamatan menggunakan lampu indikator hijau, kuning, dan merah, serta menggunakan suara peringatan menggunakan buzzer pada kondisi bahaya. Penelitian in mengabadikan area aman menggunakan kamera *monocular* dan memproses gambar *deep learning Object detector SSD MobileDets*. Untuk dapat mendeteksi objek, data orang yang telah dikumpulkan akan dilakukan proses pelatihan model, Proses pelatihan model dilakukan agar menghasilkan model yang siap untuk digunakan. Pada saat proses pelatihan, model mendapatkan nilai mAP sebesar 0,217 % pada data validasi, dan mendapatkan nilai mAP sebesar 0,274 % pada data *test*. Selanjutnya, *Detektor objek* dijalankan di *Edge TPU*, perangkat USB untuk mempercepat proses pembelajaran mendalam. Model detektor objek yang dihasilkan dapat mendeteksi objek berupa orang dengan akurasi rata-rata 73,81 %. detektor objek akan mengenali orang dan menandainya dengan kotak pembatas. Dari kotak pembatas yang dihasilkan ini, perhitungan koordinat objek diproses untuk menganalisis orang yang memasuki area aman. Saat orang terdeteksi memasuki area aman, sistem akan memberikan peringatan dengan lampu indikator yang dihasilkan. Selain itu, sistem dapat menghasilkan sinyal keamanan menggunakan *Input/Output* USB, yang dapat diintegrasikan dengan pengontrol mesin atau robot.

Kata kunci— keselamatan kerja; jaringan saraf konvolusional; pembelajaran mendalam; *Edge TPU*.

I. PENDAHULUAN

Interaksi manusia dan mesin merupakan hal yang tidak dapat terpisahkan di era Industri 4.0. Interaksi ini dapat bersifat kontak langsung ataupun secara kontak tidak langsung. Interaksi secara tidak langsung dapat dilakukan melalui teleoperasi melalui perangkat *Human-Machine Interface* (HMI). Interaksi ini biasanya dilakukan untuk operasional mesin dari jarak jauh. Sedangkan interaksi secara langsung biasanya dilakukan untuk melakukan perbaikan, dimana operator wajib hadir di area mesin. Interaksi secara langsung dengan mesin menimbulkan isu keselamatan kerja yang ditimbulkan dari sisi manusia atau mesin[1]. Isu ini dapat diklasifikasikan menjadi dua, yaitu: 1. Isu *safety internal* dan 2. Isu *safety eksternal*. Isu keselamatan internal dapat berasal dari faktor mesin, faktor manusia, ataupun dari sisi cara interaksinya. Salah satu isu keselamatan kerja yang disebabkan oleh faktor manusia adalah tingkat kelalaian manusia. Sebagai contoh,

Received xx Agxx 20xx, Revised 2x Xxxxxber 20xx, Accepted xx Xxxxber 20xx.
DOI: https://doi.org/10.15294/jte.vvolxx?inoxx?.idxx?

kelalaian operator memasuki area kerja terlarang yang di sekitarnya terdapat mesin yang sedang beroperasi. Tindakan ini dapat dicegah dengan melakukan pengembangan di sisi mesin, yaitu dengan menambahkan sensor *safety*.

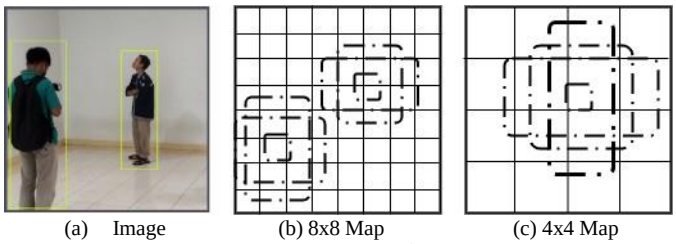
Sensor *safety* yang umum digunakan di industri adalah sensor berbasis laser, baik pemindai laser dua dimensi ataupun tiga dimensi. Sensor laser digunakan untuk mendeteksi objek bergerak yang mendekat ke area mesin. Selanjutnya, sinyal sensor diproses untuk memberikan tanda peringatan atau tanda darurat. Selain itu, hasil deteksi sensor dapat dijadikan pemicu mesin untuk berhenti secara langsung dalam keadaan darurat. Teknik pendeteksian objek bergerak ini dapat dilakukan secara langsung jika menggunakan sensor pemindai 3D (3 Dimensi) [2]. Namun, untuk sensor laser 2D (2 Dimensi), perlu dikombinasikan dengan sensor IMU *(Inertial Measurement Unit*), seperti yang telah dikembangkan oleh [3]. Walaupun sensor laser saat ini merupakan *defacto* sensor *safety* yang umum digunakan di industri, namun sensor ini masih terbilang kurang efisien dari sisi harga. Analisis ekonomi yang dilakukan oleh [4] menyebutkan bahwa harga pembelian dari sensor ini sekitar 3500 Euro. Sementara biaya lainnya yang dibebankan untuk instalasi dan pemeliharaan sensor sekitar 354.92 Euro serta terdapat biaya operasional sekitar 1.02 Euro per-hari.

Hal tersebut yang melatarbelakangi penelitian ini dilakukan. Penelitian ini dilakukan untuk mengembangkan sensor *safety* berbiaya murah sebagai alternatif pengganti sensor pemindai laser. Sensor *safety* yang dikembangkan merupakan sebuah perangkat embedded system yang terintegrasi dengan kamera. Hasil tangkapan kamera akan diproses menggunakan *deep learning* untuk mendeteksi objek bergerak. Sensor yang dikembangkan memiliki luaran sinyal digital yang dapat diintegrasikan ke *Programmable Logic Controller* (PLC) dan memiliki fitur tambahan berupa sinyal peringatan menggunakan suara.

I. METODE

A. SSD (Single Shut Multibox Detector)

SSD adalah salah satu model deteksi objek *real-time* yang populer. SSD menggunakan arsitektur *deep learning convolutional neural network* (CNN) untuk melakukan deteksi objek dengan cepat dan akurat dalam satu langkah (*single shot*). Salah satu keunggulan utama dari SSD adalah kemampuannya untuk mendeteksi objek pada berbagai skala dan aspek ratio dalam satu proses inferensi. Ini berarti SSD dapat mendeteksi objek yang berbeda dalam berbagai ukuran dan orientasi dengan efisien [5]. Pada penelitian ini menggunakan sistem deteksi objek berbasis *deep learning SSDLite-MobileDets* pada pengembangan sensor *safety*. Deteksi objek yang dilakukan menggunakan *framework Tensorflow object detection* dengan metode CNN-*Single Shot Multibox Detector* (SSD). Arsitektur *single shot multibox detector* memiliki kelebihan dari sisi model deteksinya, dimana hanya dibutuhkan satu kali proses untuk mendeteksi objek (*single shoot*)[6]. Sementara arsitektur SSD yang digunakan untuk pengembangan detektor objek adalah gabungan dari SSD dan *MobileDets*.



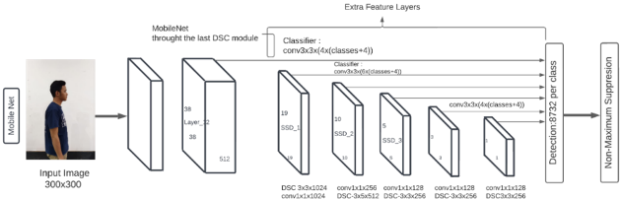
(a) Image (b) 8x8 Map (c) 4x4 Map

Gambar 1. Kotak default

Selama pelatihan, SSD memerlukan gambar-gambar sebagai *input* dan kotak-kotak pembatas (*bounding boxes*) yang menunjukkan posisi objek-objek pada gambar (kotak *ground truth*). Setiap kotak *ground truth* harus di kasih label dengan kategori objek yang tepat (misalnya, *person*) pada Gambar 1. (a) *image*. Model SSD menggunakan lapisan konvolusi untuk mengevaluasi beberapa kotak *default* pada berbagai lokasi di berbagai peta fitur dengan skala yang berbeda. Kotak *default* ini adalah kotak pembatas yang diatur sebelumnya dengan rasio aspek yang berbeda. Misalnya, pada beberapa peta fitur dengan skala yang berbeda seperti Gambar 1. (b) 8x8 map dan Gambar 1. (c) 4x4 map, SSD akan mengevaluasi kotak-kotak *default* ini. Untuk setiap kotak *default*, SSD memprediksi dua hal: pergeseran bentuk (lokalisasi) dan konfidensi untuk setiap kategori objek. Dengan kata lain, model mencoba untuk memahami seberapa baik kotak *default* tersebut cocok dengan objek sebenarnya dan seberapa besar kemungkinan ada objek dalam kotak tersebut. Misalnya, jika ada kotak *default* yang seharusnya menutupi gambar *person*, model akan mencoba memprediksi sejauh mana kotak tersebut perlu digeser atau diubah bentuknya untuk mencocokkan dengan gambar *person*, serta seberapa besar kemungkinan bahwa itu adalah *person*. model mencocokkan kotak-kotak *default* yang telah dievaluasi dengan kotak *ground truth* yang sesuai dengan objek yang sebenarnya. Misalnya, jika dua kotak *default* cocok dengan kucing dan satu kotak cocok dengan anjing pada gambar, kotak-kotak tersebut akan dianggap sebagai positif, sementara kotak-kotak lainnya dianggap sebagai negatif.

Kerugian (*loss*) model dihitung dengan cara menggabungkan dua komponen utama: kerugian lokalisasi dan kerugian kepercayaan. Kerugian lokalisasi mengukur sejauh mana kotak-kotak *default* yang diprediksi cocok dengan kotak *ground truth* dalam hal pergeseran bentuk. Kerugian kepercayaan mengukur seberapa baik model memprediksi kategori objek yang benar. Biasanya, kerugian lokalisasi dihitung menggunakan metode seperti *Smooth* L1, sementara kerugian kepercayaan menggunakan *Softmax*. Dengan menggabungkan kedua komponen ini, model SSD diberi umpan balik selama pelatihan untuk memperbaiki prediksi lokalisasi dan prediksi kategori objek. Tujuan akhirnya adalah untuk menghasilkan model yang dapat mendeteksi objek dengan cepat dan akurat dalam satu langkah *(single shot*) pada berbagai skala dan aspek ratio.

Jaringan dasar SSD menggunakan jaringan VGG16. Dua lapisan yang terhubung sepenuhnya FC6 dan FC7 dari VGG16 digantikan oleh lapisan konvolusi. Semua lapisan *Dropout* dan Lapisan FC8 dihapus. Empat lapisan konvolusi Conv8_2, Conv9_2, Conv10_2, dan Conv11_2 ditambahkan setelah VGG16 yang dimodifikasi. Lapisan konvolusi baru digunakan untuk mendapatkan lebih banyak fitur untuk pendeteksian[7]. Jaringan SSD Struktur jaringan SSD seperti yang ditunjukkan pada gambar 2.



Gambar 2. Arsitekstur SSD

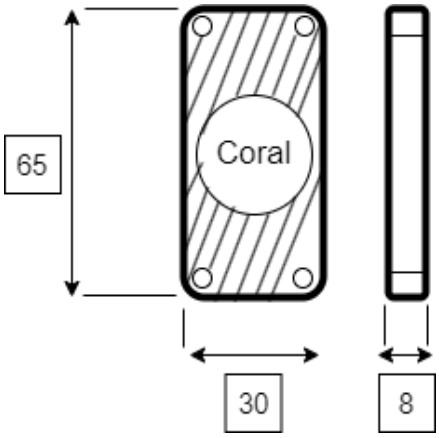
Arsitektur SSD menggabungkan pendekatan *Region Proposal Networks* (RPN) dan detektor berbasis *grid*. Pertama, lapisan konvolusi digunakan untuk ekstraksi fitur dari gambar *input*. Selanjutnya, setiap sel pada lapisan fitur tersebut mengevaluasi beberapa *bounding box* dengan skala dan *aspect* ratio yang berbeda (disebut kotak pembungkus atau "*multibox*")[8]. Setiap kotak pembungkus mengandung informasi tentang kelas objek yang berbeda dan *offset* untuk memperbaiki posisi kotak tersebut. SSD memiliki beberapa lapisan deteksi pada berbagai tingkat resolusi lapisan fitur (misalnya, lapisan dengan tingkat resolusi rendah untuk deteksi objek besar dan lapisan dengan tingkat resolusi tinggi untuk deteksi objek kecil). Hasil akhirnya adalah kombinasi dari semua deteksi dari berbagai lapisan, yang memberikan kemampuan deteksi objek pada berbagai ukuran dan skala.

B. MobileDets

MobileDets adalah pendekatan yang dikembangkan untuk mengoptimalkan deteksi objek pada perangkat bergerak (*mobile devices*) dengan keterbatasan sumber daya komputasi. *MobileDets* mencoba untuk menemukan arsitektur deteksi objek yang efisien dengan mempertimbangkan batasan daya komputasi pada perangkat bergerak. Untuk itu, mereka menggunakan teknik seperti *search space* yang dioptimalkan untuk perangkat seluler, serta pencarian arsitektur secara otomatis (*AutoML*) untuk menemukan arsitektur yang terbaik dengan kompromi antara kecepatan dan akurasi deteksi[9]. *MobileDets* merupakan arsitektur CNN (*Convolutional Neural Network*) yang dioptimalkan untuk penggunaan perangkat *embedded system* atau seluler. Ini membantu meningkatkan *trade-off* akurasi-*latency* untuk deteksi objek pada akselerator, asalkan mereka ditempatkan secara strategis di jaringan melalui pencarian arsitektur saraf [10]. Dengan menggabungkan konvolusi reguler di ruang pencarian dan langsung mengoptimalkan arsitektur jaringan untuk deteksi objek. Sehingga jika dibandingkan dengan *baseline* model dari SSD, *SSDLite-MobileDets* memiliki kecepatan komputasi yang lebih cepat.

C. Coral Accelerator

Coral Accelerator, atau yang juga dikenal sebagai *Edge* TPU (*Tensor Processing Unit*) adalah *chip hardware* akselerator khusus yang dikembangkan oleh Google [11]. Tujuan dari *Coral Accelerator* adalah untuk memungkinkan pelaksanaan inferensi (pengambilan prediksi) model *machine learning* dengan kecepatan tinggi dan konsumsi daya yang rendah pada perangkat *edge* atau perangkat yang terhubung ke *Internet of Things* (IoT). *Coral Accelerator* didasarkan pada arsitektur *Tensor Processing Unit* (TPU) khusus Google. Dengan konsumsi daya hanya dua watt, *Edge* TPU sangat sesuai untuk lingkungan berdaya rendah. Secara khusus, pada jaringan saraf dalam yang dipilih, *Edge* TPU mengungguli akselerator perangkat keras lainnya saat mengukur gambar per detik per Watt. Setiap IC *Edge* TPU dapat beroperasi pada empat triliun operasi per detik (TOPS), menghasilkan dua TOPS per Watt. Saat ini, beberapa *platform* perangkat keras tersedia untuk tujuan pembuatan prototipe dan produksi. Dengan *platform* Coral, Google menyediakan perpustakaan model yang komprehensif yang dapat langsung digunakan[12]. TPU adalah chip desain khusus yang dioptimalkan untuk melakukan operasi tensor dan komputasi paralel yang diperlukan dalam proses *deep learning*. TPU dirancang untuk mempercepat operasi matriks dan tensor yang merupakan komponen utama dari model *neural network*. *Coral Accelerator* menyediakan performa tinggi dengan daya yang rendah, membuatnya sangat cocok untuk perangkat *edge* seperti perangkat IoT, kamera pintar, sistem keamanan, dan lain sebagainya. Dengan menggunakan chip akselerator ini, inferensi model *deep learning* dapat dijalankan secara cepat dan efisien tanpa memerlukan sumber daya komputasi yang besar[13]. Hal ini membuka jalan untuk melakukan deteksi objek yang akurat, murah, dan cepat, bahkan cocok untuk aplikasi industri dalam Industri 4.0. Empat jaringan *neural* canggih dilatih melalui pembelajaran transfer, kemudian diterapkan dan diuji pada *Raspberry Pi 4B* dan akselerator TPU *Coral Edge* dari Google sebagai *co-processor*. Perbandingan model-model tersebut berfokus pada waktu inferensi, fleksibilitas penerapan, pelatihan, dan akhirnya akurasi jaringan yang telah dilatih ulang pada sejumlah dataset dengan karakteristik fitur yang berbeda[14].



Gambar 3. Coral Accelerator

Terlihat pada Gambar 3. *Coral Accelerator* adalah perangkat USB (*Universal Serial Bus*) yang menawarkan kemampuan *Machine Learning* (ML) yang sangat cocok unruk sistem Linux. Dikombinasikan dengan *Edge* TPU (*Tensor Proccesing Unit*), yang merupakan ASIC (*Application Specific Integrated Circuit*) kecil yang dirancang oleh Google, perangkat ini dapat memberikan pembelajaran mesin maksimum dengan kinerja tinggi tetapi tetap dengan konsumsi daya rendah melalui antarmuka USB 3.0.

D. Tensorflow Objek *Detection* API

TensorFlow adalah sebuah *library* (pustaka) sumber terbuka (*open-source*) yang fokus pada kecerdasan buatan (AI) dan pembelajaran mesin (*Machine Learning*). *Library* ini dikembangkan oleh tim Google Brain dan dirilis pertama kali pada tahun 2015. TensorFlow merepresentasikan komputasi dengan grafik aliran data[15]. Meskipun berfokus pada aplikasi pembelajaran mesin, TensorFlow agak agnostik pada tujuan yang tepat dari perhitungan. Secara khusus, komputasi dapat melakukan satu atau lebih langkah pelatihan untuk model pembelajaran mesin, atau mungkin penerapan model yang sudah dilatih [16].

Model TensorFlow dirancang untuk menyediakan kerangka kerja yang kuat dan efisien untuk membangun, melatih, dan menerapkan model jaringan saraf tiruan (*neural network*) serta melakukan berbagai komputasi numerik yang kompleks. Pada tugas akhir ini menggunakan tensorflow versi 1.15. TensorFlow adalah *library* yang sangat kuat dan dapat digunakan untuk mengoptimalkan, melatih, dan menerapkan model deteksi objek. Salah satu keunggulan TensorFlow adalah kemampuannya untuk membangun model pembelajaran mesin

dengan menggunakan data pelatihan besar, seperti dataset *Common Contextual Objects* (COCO). Dataset COCO adalah salah satu dataset paling populer untuk pelatihan model deteksi objek. Dataset ini terdiri dari sekitar 300.000 gambar yang mencakup 90 jenis objek yang berbeda. Setiap objek pada gambar diberi label, sehingga model dapat belajar untuk mengenali dan mendeteksi objek-objek ini dalam gambar baru. API TensorFlow yang digunakan untuk deteksi objek didukung oleh berbagai model yang berbeda, dan setiap model memiliki keseimbangan antara kecepatan dan akurasi. Dengan API yang dipikirkan dengan matang, Keras sangat populer di kalangan pemula *deep learning* dan praktisi ML terapan. Inti dari API adalah konsep model dan lapisan. Pengguna dapat membangun sebuah model dengan mengumpulkan satu set lapisan yang telah ditentukan sebelumnya, di mana setiap lapisan memiliki *default* yang wajar parameter yang wajar untuk mengurangi beban kognitif[17].

E. Jupyter Notebook

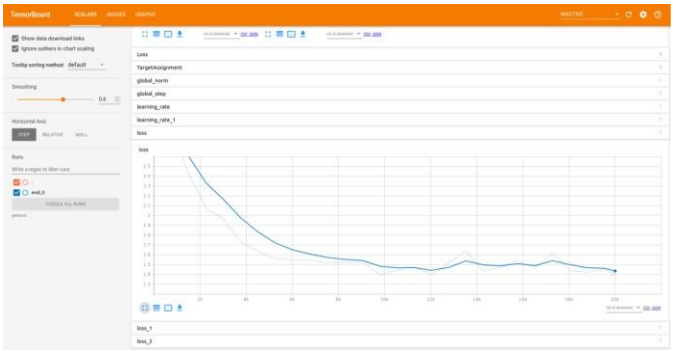
Jupyter Notebook adalah sebuah lingkungan pengembangan interaktif yang memungkinkan para peneliti, data *scientist*, dan pengembang untuk bekerja dengan kode, visualisasi, dan teks dalam satu tempat. Ini adalah salah satu alat yang paling populer dalam komunitas ilmu data karena kemampuannya untuk memadukan kode, hasil eksekusi, serta analisis atau penjelasan dalam satu dokumen yang dapat dibagikan dan direproduksi serta dapat diakses melalui peramban web[18]. Dalam *Jupyter Notebook*, para pengguna dapat mengorganisir kode ke dalam sel-sel yang dapat dijalankan secara independen atau berurutan. Hal ini memungkinkan mereka untuk mengeksplorasi data dan melakukan percobaan iteratif dengan lebih mudah, karena setiap sel dapat dieksekusi dan hasilnya ditampilkan di tempat. Selain itu, *Jupyter Notebook* mendukung banyak bahasa pemrograman, termasuk Python, R, Julia, dan lainnya, sehingga para pengguna dapat memilih bahasa yang sesuai dengan kebutuhan analisis mereka[19]. Dalam tugas akhir ini, menggunakan *Jupyter Notebook* untuk mengeksekusi program python untuk pelatihan model, pengujian model pelatihan, dan melakukan konversi model agar model dapat dijalankan pada *Edge* TPU. Hal yang hebat tentang *Jupyter Notebook* adalah tampilan nya yang cukup rumit dan teknis, tetapi pada kenyataannya tidak terlalu sulit untuk digunakan. Secara keseluruhan, ini adalah alat yang hebat untuk digunakan untuk semua pemrograman. Tidak hanya itu, ini juga dapat digunakan untuk menampilkan hasil dengan cara yang tidak terlalu berat untuk dilihat.

F. TensorBoard

TensorBoard adalah sebuah alat visualisasi yang dikembangkan oleh Google untuk mempermudah pemahaman, pemantauan, dan analisis model deep learning yang dibangun dengan menggunakan *framework* TensorFlow. Ini memungkinkan pengembang dan peneliti untuk melihat dengan lebih jelas bagaimana model mereka bekerja, melacak metrik kinerja selama pelatihan, menganalisis grafik komputasi, dan melakukan berbagai tugas visualisasi lainnya. *TensorBoard* juga memfasilitasi proses *debugging* dan tuning model *deep learning*.

Visualisasi Grafik Komputasi: *TensorBoard* memungkinkan kita untuk melihat grafik komputasi model dalam bentuk grafik berarah (*directed graph*), yang dapat membantu dalam memahami arsitektur model dengan lebih baik[20]. Monitoring Metrik: kita dapat melacak metrik seperti *loss*, akurasi, dan metrik evaluasi lainnya selama pelatihan model. Ini dapat membantu memantau kinerja model selama waktu. Visualisasi Data: kita dapat menggunakan TensorBoard untuk memvisualisasikan data *input* dan *output* model, serta berbagai statistik terkait data. Histogram dan Distribusi: Ini memungkinkan kita untuk memahami bagaimana bobot dan bias dalam model Anda berubah selama pelatihan dengan melihat histogram distribusi mereka. Profil Kinerja: kita dapat melakukan profil kinerja model untuk mengidentifikasi *bottleneck* dalam pelatihan atau inferensi. *Embedding Projector*: *TensorBoard* juga memiliki fitur yang memungkinkan memvisualisasikan representasi data bertingkat (misalnya, *embedding word vectors*) dalam tampilan 3D interaktif. *Plugin* dan Ekstensi: *TensorBoard* dapat diperluas dengan *plugin* dan ekstensi yang dapat disesuaikan untuk memenuhi kebutuhan.

Model jaringan saraf sering kali merupakan struktur yang besar dan berbelit-belit dan dapat menyebabkan kebingungan. *TensorBoard* adalah alat visualisasi yang membuatnya lebih mudah dipahami dan men-*debug* program TensorFlow. Pengguna dapat menggunakan *TensorBoard* untuk memvisualisasikan dengan mudah grafik komputasi model, metrik pelatihan, dan nilai parameter. Modul TensorFlow, *tf.summary*, menciptakan operasi untuk menyimpan definisi grafik dan mencatat statistik pelatihan. Modul ini juga menuliskannya sebagai file peristiwa ke *hard disk* dalam format yang dapat dibaca oleh *TensorBoard*. File-file ini akan divisualisasikan di browser ketika *TensorBoard* diluncurkan.



Gambar 4. TensorBoard

Terlihat pada Gambar 4. Penelitian ini menggunakan *TensorBoard* untuk memonitor kemajuan pelatihan untuk mengenali ketika masalah seperti *overfitting* terjadi. Dasbor *Scalars* menampilkan perkembangan kerugian dan metrik seiring berjalannya waktu. *Tensorboard* pada penelitian ini dapat digunakan untuk memantau perkembangan pelatihan, perubahan tingkat pembelajaran, dan pengukuran lainnya. Dasbor Grafik menunjukkan struktur model. Sementara itu, Dasbor Distribusi dan Histogram menghadirkan visualisasi dari distribusi tensor sepanjang rangkaian waktu [21]. Informasi ini dapat sangat berguna untuk melihat perkembangan bobot dan bias, serta memastikan bahwa mereka berkembang sesuai dengan yang diharapkan. Memanfaatkan *TensorBoard* untuk merinci metrik pelatihan, seperti kerugian dan akurasi dalam pelatihan dan validasi, memberikan kita kemampuan untuk mengidentifikasi apakah kita mengalami *overfitting*, *underfitting*, atau berada dalam kondisi yang optimal.

I. TAHAP PELAKSANAAN

A. Pengumpulan Data Pelatihan

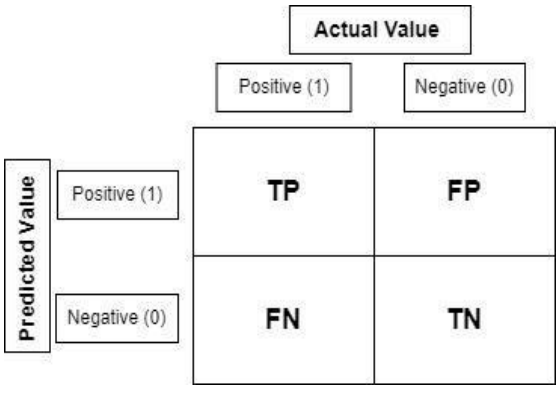
Dalam melakukan sebuah pelatihan, diperlukan data pelatihan yang nantinya akan dilatih menggunakan algoritma yang telah ditentukan. Data tersebut berisi gambar dan anotasi (label) yang sesuai dengan objek yang ingin di deteksi dalam gambar, dan data tersebut juga akan dikonversi sesuai format yang akan di proses pada saat pelatihan model. Dalam penelitian ini, data yang dikumpulkan sebanyak 5000 gambar yang telah dianotasi, gambar yang telah dianotasi ini dibagi menjadi 3 bagian, yaitu gambar pelatihan, gambar validasi dan gambar evaluasi. Gambar yang digunakan untuk proses pelatihan sebanyak 4000 gambar, sedangkan untuk validasi sebanyak 700 gambar, dan untuk evaluasi sebanyak 300 gambar. Selanjutnya, data yang telah dianotasi akan konversi ke format *Tfrecord*, dimana format data tersebut yang akan diproses oleh algoritma *SSDLite-MobileDets*.

B. Pelatihan Model Objek *Detector*

Pada proses pelatihan model, *SSDLite-MobileDets* menggunakan *pre-trained* model (model yang telah dilatih pada dataset *ImageNet*) sebagai dasar. Pelatihan ini dapat mengimpor model *pre-trained* dan kemudian menggantinya sesuai dengan kebutuhan. Sebelum melakukan pelatihan model, kita perlu melakukan konfigurasi config. Konfigurasi config adalah definisi dari file konfigurasi yang akan digunakan nanti untuk mengonfigurasi model pelatihan[22]. Ada beberapa bagian yang harus disesuaikan dengan model yang akan dilatih :

1) Menyesuaikan dengan jumlah kelas yang digunakan.
2) Mengubah letak *fine_tune_checkpoint*.
3) Mengatur jumlah *step* yang akan digunakan pada saat pelatihan.
4) Mengarahkan direktori ke lokasi file *train.record* dan label_map.pbtxt
5) Mengubah *num_example*.

Proses pelatihan adalah proses melakukan pelatihan pada data yang telah dikumpulkan. Dari algoritma-algoritma yang telah ditentukan akan menghasilkan model deteksi. Model pelatihan menggunakan CNN (*Convolution Neural Network*). *Mean Average Precision* (mAP) adalah metrik tolok ukur saat ini yang digunakan untuk mengevaluasi kekokohan model deteksi objek. mAP merangkum *tradeoff* antara presisi dan recall dan memaksimalkan efek dari kedua metrik. Positif benar dan salah tugas deteksi objek diklasifikasikan menggunakan ambang batas *Intersection over Union* (IoU). Menghitung mAP pada rentang ambang batas IoU menghindari ambiguitas dalam memilih ambang batas IoU yang optimal untuk mengevaluasi akurasi model[23].



Gambar 5. Confusion matrix

Terlihat pada Gambar 5. *Confusion matrix* digunakan untuk memperoleh nilai *accuracy*, *recall*, dan *precision*. Berikut adalah parameter-parameter yang digunakan untuk menentukan nilai-nilai tersebut:

1) True Positive (TP) adalah ketika nilai sebenarnya dinyatakan Positif dan diprediksi juga sebagai Positif.
2) True Negatif (TN) adalah ketika nilai sebenarnya dinyatakan Negatif dan prediksi juga sebagai Negatif.
3) False Positive (FP) adalah Ketika nilai sebenarnya Negatif tetapi diprediksi sebagai Positif.
4) False Negative (FN) adalah Ketika nilai sebenarnya Positif tetapi diprediksi sebagai Negatif.
5) *Accuracy* adalah perhitungan total seberapa akurat objek di seluruh *frame* saat menjalani model deteksi. Rumus perhitungan *accuracy* dapat dilihat pada persamaan (1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Accuracy = \frac{Total\ Prediksi\ benar}{Total\ semua\ prediksi} \quad (1)$$

6) *Precision* adalah total semua prediksi yang benar dibandingkan semua hasil yang terprediksi oleh sistem. Dalam parameter ini *precision* akan menentukan berapa jumlah objek yang dinyatakan benar dari semua jumlah objek yang dinyatakan oleh sistem. Rumus perhitungan *precision* dapat dilihat pada persamaan (2).

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Precision = \frac{Prediksi\ Dinyatakan\ Positif}{Total\ Prediksi\ Dinyatakn\ Positif} \quad (2)$$

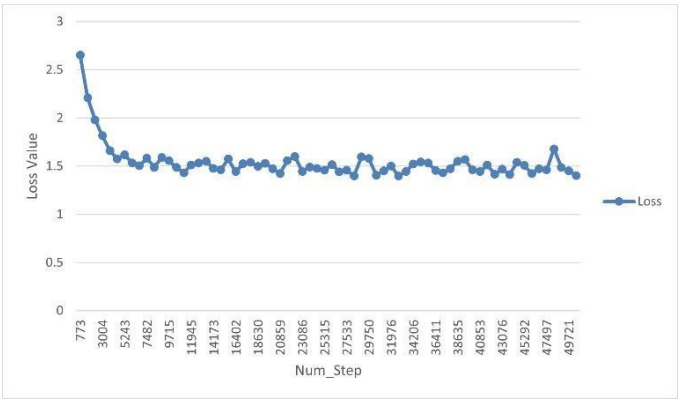
7) *Recall* adalah total estimasi yang benar dibandingkan dengan semua hasil yang sebenarnya. *Recall* menjadi parameter penentu objek yang akan dinyatakan benar dari keseluruhan objek sebenarnya. Perhitungan *recall* dapat dilihat pada persamaan (3).

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$Recall = \frac{Prediksi\ Dinyatakan\ Positif}{Total\ Aktual\ Dinyatakan\ Positif} \quad (3)$$

Fungsi *Loss* atau fungsi biaya digunakan untuk mengukur kesalahan antara nilai prediksi dan nilai aktual. Biasanya, fungsi kerugian digunakan sebagai kriteria pembelajaran dalam masalah optimasi. Dalam konteks CNN, berbagai jenis fungsi kerugian dapat digunakan untuk menangani masalah regresi dan klasifikasi. Tujuannya adalah untuk meminimalkan kesalahan prediksi. Salah satu fungsi kerugian yang umum digunakan dalam tugas klasifikasi, disebut kerugian entropi silang, mengevaluasi perbedaan antara distribusi probabilitas yang diprediksi selama pelatihan dan distribusi aktual. Fungsi ini membandingkan probabilitas prediksi dengan nilai *output* aktual (0 atau 1) di setiap kelas dan menghitung penalti berdasarkan jarak di antara keduanya. Penalti ini bersifat logaritmik, sehingga fungsi ini memberikan nilai yang lebih

rendah (0,1 atau 0,2) untuk perbedaan kecil dan skor yang lebih tinggi (0,9 atau 1,0) untuk perbedaan yang lebih besar[24].



Gambar 6. Grafik pelatihan model

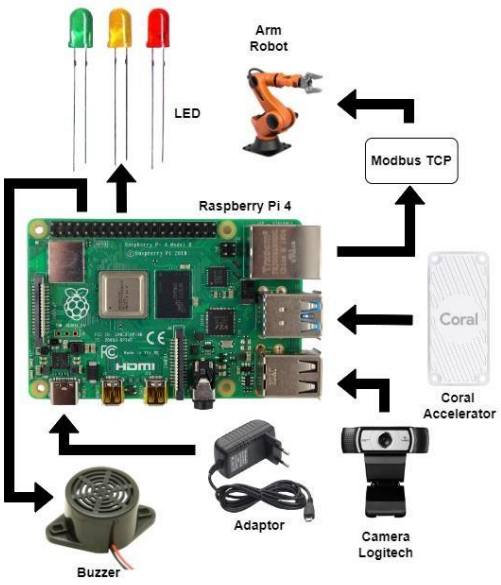
Gambar 6. Adalah grafik nilai *Loss* pada proses pelatihan model. Pada pelatihan model yang dilakukan menggunakan 5000 data gambar mendapatkan nilai *Loss* akhir sebesar 1,47. Proses pelatihan data ini bertujuan untuk meminimalkan kesalahan prediksi terhadap data latih. Setelah melakukan pelatihan data secara iteratif dan mendapatkan kesalahan prediksi yang minimal, selanjutnya nilai parameter dan bias dari *SSDLite-MobileDets* diambil dan disimpan pada sebuah berkas dengan ekstensi .ckpt, untuk selanjutnya dioptimalkan dan digunakan pada fase inferensi.

C. Kuantisasi Model Pascal Pelatihan

Setelah mendapatkan nilai *weights* dan *biases* dari model *SSDLite-MobileDets*, selanjutnya diperlukan proses optimasi terhadap ukuran parameter tersebut. Proses optimasi ini bertujuan untuk memperkecil ukuran *weights* dan *biases* yang diperoleh, tanpa mengurangi nilai akurasi. Semakin kecil nilai *weights* dan *biases*, maka semakin cepat sistem deteksi objek bekerja. Proses optimasi ini dapat dilakukan dengan kuantisasi data, atau mengubah representasi data ke lebar data yang lebih rendah. Hasil optimasi ini nantinya akan disimpan pada sebuah berkas dengan ekstensi.*Tflite*. Pada pengembangan sensor pemindai ini menggunakan berkas tersebut pada *embedded computer Raspberry Pi* yang dilengkapi dengan Google *Coral* TPU sebagai perangkat akselerator deteksi objek [25].

D. Perangkat Keras

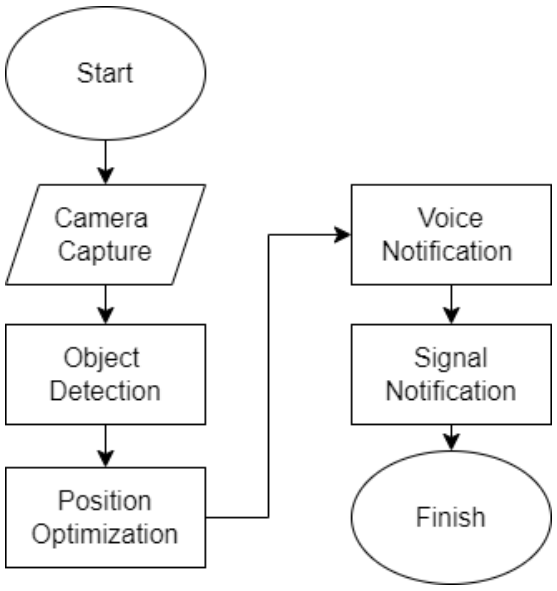
Rancangan perangkat keras yang diusulkan pada penelitian ini divisualisasikan pada Gambar 7. Rancangan perangkat keras terdiri dari beberapa komponen utama, yaitu: 1. *Raspberry Pi*, 2. Kamera, 3. *Coral* USB *Accelerator* dan 4. Relay. Kamera digunakan sebagai sensor utama untuk mendeteksi objek. *Raspberry Pi* digunakan sebagai perangkat *embedded system* untuk implementasi algoritma. Sedangkan kamera akan menangkap gambar di sekitar untuk selanjutnya di proses oleh algoritma deteksi objek. Karena algoritma deteksi objek yang digunakan berbasis *deep learning*, maka pemrosesan di *processor* saja tidak cukup. Maka di rancangan sistem ini digunakan *Coral* USB *Accelerator* sebagai perangkat untuk melakukan inferensi deteksi objek. Sementara Relay berfungsi luaran sinyal digital yang dapat dihubungkan dengan mesin ataupun perangkat lainnya.



Gambar 7. Rancangan perangkat keras

E. Perangkat Lunak

Rancangan perangkat lunak pada penelitian ini dideskripsikan oleh diagram alir pada Gambar 8. Alur kerja perangkat lunak pada sistem dimulai dari akuisisi gambar dari kamera. Gambar tersebut kemudian akan diproses menggunakan algoritma deteksi objek. Hasil deteksi ini akan menghasilkan estimasi lokasi objek pada koordinat gambar. Hasil ini kemudian diproses lebih lanjut untuk mendapatkan estimasi lokasi objek terhadap area yang telah ditentukan. Setelah itu, data koordinat Y hasil deteksi akan diproses untuk mendapatkan peringatan berupa lampu peringatan hijau, kuning, dan merah. Lampu hijau adalah kondisi dimana objek berada di area yang aman. Lampu kuning mengidentifikasi objek berada di area peringatan, dan lampu merah mengidentifikasi objek berada di area bahaya yang dapat menyebabkan dan mengancam operator mengalami kecelakaan kerja di lingkungan *hazard*.



Gambar 8. Rancangan perangkat lunak

I. HASIL DAN PEMBAHASAN

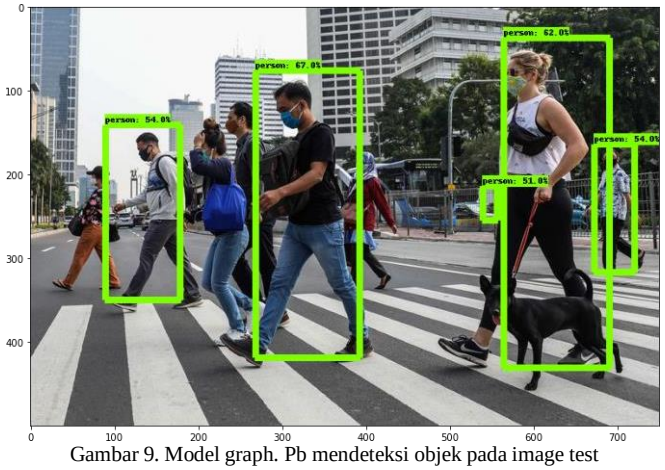
A. Kinerja Detektor Objek

Analisis kinerja detektor objek dilakukan untuk mengetahui tingkat akurasi dari *SSDLite-MobileDets*. Untuk analisis kinerja ini digunakan metrik evaluasi *mean Average Precission* (mAP). Analisis kinerja dilakukan sebelum dan sesudah proses optimasi *weights* dan *biases*. Analisis ini dilakukan dengan tujuan untuk mengetahui penurunan performa detektor objek saat implementasi langsung ke perangkat *embedded system*. Selain melakukan analisis terhadap mAP, analisis juga dilakukan pada kecepatan deteksi pada *Coral* USB *Accelerator*. Analisis dilakukan dengan cara mengukur kecepatan deteksi dalam *frame per second* (FPS). Analisis ini juga dilakukan sebelum dan sesudah proses optimasi *weights* dan *biases*, dengan tujuan untuk mengukur penurunan atau peningkatan kecepatan deteksi.

TABEL I. MEAN AVERAGE PRECISION

No	Arsitekstur CNN	mAP@0.5 (%) Validation (%)	mAP@0.5 (%) Test (%)
1	SSD-MobileDet-TPU	0,217	0,274
2	SSD-MobileDet-CPU	0,260	0,258

Pada analisis mAP (*Mean Average Precision*) yang dilakukan menggunakan data pelatihan validasi dan *test*. Terlihat pada tabel 1, pada arsitekstur *SSD-MobileDet-TPU* nilai mAP untuk data validasi sebesar 0,217 %, dan untuk data *test* sebesar 0,274 %. Untuk arsitekstur SSD-MobileDet-CPU, nilai mAP untuk data validasi sebesar 0,260 %, dan untuk data *test* sebesar 0,258 %.



Gambar 9. Model graph. Pb mendeteksi objek pada image test

Pada gambar 9 diatas merupakan proses pengujian model *graph.pb* yang telah dikonversi dari model *checkpoint* dari hasil pelatihan. Pengujian tersebut dilakukan menggunakan *image test*. Hasilnya model tersebut berhasil mendeteksi beberapa *person* pada *image test* dengan akurasi rata-rata mencapai 57,6 %.

Gambar 10. Model mendeteksi objek berupa person

Untuk melakukan inferensi pada *Coral Edge* TPU, model hasil pelatihan perlu di konversi ke format *person_edgetpu.tflite*. Terlihat pada Gambar 10. hasilnya model tersebut berhasil mendeteksi *person* dengan akurasi rata-rata 72,28 %.

TABEL II. DETECTION PERFORMANCE

No	Accuracy	FPS	Inference (ms)	Condition
1	74.61 %	17.17 fps	152.15 ms	Warning
2	75.78 %	14,37 fps	130.90 ms	Warning
3	75.78 %	20.74 fps	136.77 ms	Warning
4	77.73 %	25.94 fps	120.88 ms	Warning
5	77.73 %	21.22 fps	116.32 ms	Warning
6	77.73 %	18.90 fps	136.58 ms	Warning
7	77.73 %	17.27 fps	132.77 ms	Warning
8	77.73 %	19.62 fps	135.86 ms	Warning
9	77.73 %	18.09 fps	157.35 ms	Warning
10	77.78 %	26.97 fps	132.32 ms	Warning
11	76.56 %	27.31 fps	106.32 ms	Danger
12	76.56 %	25.76 fps	90.00 ms	Danger
13	77.73 %	30.48 fps	85.83 ms	Danger
14	77.73 %	30.57 fps	85.39 ms	Danger
15	76.56 %	28.05 fps	88.75 ms	Danger
16	76.56 %	27.55 fps	89.67 ms	Danger
17	76.56 %	26.74 fps	92.36 ms	Danger
18	76.56 %	26.79 fps	88.36 ms	Danger
19	76.56 %	26.15 fps	94.42 ms	Danger
20	76.56 %	29.90 fps	89.18 ms	Danger
21	68.36 %	29.13 fps	105.18 ms	Safe
22	69.92 %	20.88 fps	119.66 ms	Safe
23	67.19 %	25.71 fps	96.52 ms	Safe
24	67.19 %	24.76 fps	90.90 ms	Safe
25	68.36 %	15.52 fps	124.17 ms	Safe
26	64.45 %	35.68 fps	97.61 ms	Safe
27	64.45 %	29.96 fps	84.16 ms	Safe
28	64.45 %	32.67 fps	98.03 ms	Safe
29	68.36 %	29.93 fps	109.59 ms	Safe
30	73.44 %	22.14 fps	108.05 ms	Safe

Dari tabel 2, menunjukkan bahwa *Raspberry Pi 4B* menggunakan *Coral Accelerator* memiliki kinerja yang baik dalam mendeteksi objek berupa kelas *person* dalam berbagai kondisi. Pada kondisi *safe* model berhasil mendeteksi objek *person* dengan akurasi rata-rata mencapai 67.62 %, jumlah *frame* per detik rata-rata mencapai 26.63 fps, dan latensi rata-

rata mencapai 103.38 ms sedangkan pada kondisi *warning* model berhasil mendeteksi objek *person* dengan akurasi rata-rata mencapai 77.03 %, jumlah *frame* per detik rata-rata mencapai 20.65 fps, dan latensi rata-rata mencapai 135.19 ms dan pada kondisi *danger* model berhasil mendeteksi objek *person* dengan akurasi rata-rata mencapai 76.79 %, jumlah *frame* per detik rata-rata mencapai 27.93 fps, dan latensi rata-rata mencapai 91.02 ms.

TABEL III. HARDWARE PERFORMANCE

No	Prosesor	FPS	Inferensi (ms)
1	CPU (AMD-A4-9125)	3,34 fps	222,5 ms
2	TPU (Coral Accelerator)	48,8 fps	38,52 ms

Dari tabel 3, terlihat bahwa *Coral Accelerator* dengan TPU memiliki kinerja yang jauh lebih baik dibandingkan dengan CPU AMD-A4-9125 dalam konteks tugas yang diukur (dalam hal *frame* per detik). TPU menunjukkan kemampuan untuk menghasilkan 48,8 fps, sementara CPU hanya mampu menghasilkan 3,34 fps. Selain itu, latensi (waktu yang dibutuhkan untuk memproses satu *frame*) TPU juga jauh lebih rendah (38,52 ms) dibandingkan dengan CPU (222,5 ms), menunjukkan bahwa TPU lebih efisien dan responsif dalam menangani tugas-tugas dalam AI dengan cepat dan menjadikannya solusi yang populer untuk aplikasi kecerdasan buatan di *edge devices*, di mana sumber daya komputasi terbatas, dan respons waktu yang cepat dibutuhkan.

B. Sinyal Peringatan

Analisis sinyal peringatan dilakukan untuk mengetahui tingkat kesuksesan hasil sinyal yang diberikan oleh sistem. Analisis dilakukan dengan cara mencoba memasuki area *safe*, area *warning* dan area *danger*. Kemudian membandingkan sinyal luaran yang dihasilkan, baik berupa sinyal suara dan sinyal digital *output*. Analisis fungsional dilakukan untuk menguji sistem secara keseluruhan. Parameter untuk menentukan batas *area safe*, *warning*, dan *danger* menggunakan nilai koordinat Y *bounding box* pada deteksi objek. Sistem menggunakan pin GPIO pada *Raspberry Pi 4B* untuk menghasilkan sinyal peringatan berupa lampu indikator berwarna hijau, kuning, dan merah, serta menggunakan buzzer untuk suara peringatan jika memasuki area *danger*.

Gambar 11. Kondisi safe (aman)

Terlihat pada Gambar 11. Sistem memberikan sinyal peringatan *safe* menggunakan tampilan UI (*User Interface*) pada LCD *Raspberry Pi 4B* disertai dengan lampu indikator hijau, karena operator atau *person* masuk ke area yang aman pada lingkungan kerja.

Gambar 12. Kondisi warning (peringatan)

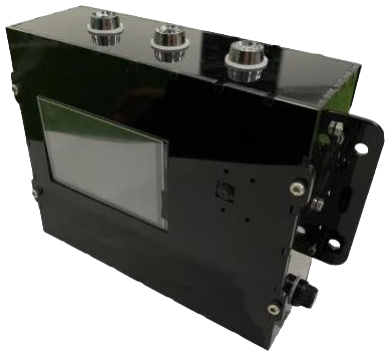
Terlihat pada Gambar 12. Sistem memberikan sinyal peringatan *warning* disertai dengan lampu indikator kuning, karena operator atau *person* masuk ke area peringatan pada lingkungan kerja.

Gambar 13. Kondisi danger (bahaya)

Terlihat pada gambar 13. Sistem memberikan sinyal peringatan *danger*, disertai dengan lampu indikator merah, dan alarm peringatan menggunakan buzzer, karena operator atau *person* masuk ke area yang berbahaya pada lingkungan kerja.

C. Fungsional Sensor Pemindai

Pada analisis fungsional, penelitian ini merencanakan untuk mengintegrasikan sistem yang dikembangkan dengan robot manipulator yang ada di lab robotika Politeknik Negeri Batam. Sinyal *output* pada sistem pemindai akan dihubungkan secara langsung pada *safety input* pada pengendali robot manipulator. Sebagai uji coba robot akan dioperasikan secara otonom menggunakan program yang telah didefinisikan. Dalam uji coba tersebut, robot manipulator akan otomatis berhenti beroperasi jika ada orang mendekat atau memasuki area berbahaya. Selain itu, sistem pemindai juga memberikan peringatan melalui lampu kuning (peringatan) dan lampu hijau (aman). Warna lampu yang muncul ditentukan berdasarkan posisi orang terhadap robot manipulator atau hasil perhitungan dari koordinat *bounding box* Y pada deteksi objek.



Gambar 14. Prototipe sistem pemindai

Pada gambar 14. Terlihat bentuk dari sistem pemindai yang dihasilkan. Pada sistem tersebut terdapat 3 buah lampu indikator hijau, kuning, dan merah, *Raspberry Pi 4B*, LCD dengan ukuran 3.5 inci, kamera, relay, *Coral Accelerator*, dan buzzer. Sistem pemindai ini dirancang secara portabel untuk memudahkan perpindahan dan penempatan di area *hazard* pada lingkungan industri.

I. KESIMPULAN

Pada penelitian tugas akhir ini menghasilkan, dan mengembangkan sensor *safety* dengan biaya yang rendah, sensor *safety* yang dihasilkan dapat memberikan sinyal peringatan berupa 3 kondisi, yaitu kondisi *safe* (aman), *warning* (peringatan), dan *danger* (bahaya) dari proses pendeteksian orang menggunakan kalkulasi nilai koordinat Y *bounding box* pada deteksi objek. Inferensi model deteksi yang dilakukan menggunakan Coral *Edge* TPU menghasilkan nilai akurasi rata-rata mencapai 73,81 %, dengan nilai FPS rata-rata mencapai 25.22 fps, serta inferensi rata-rata mencapai 109.86 ms. Dengan hasil tersebut model deteksi akan lebih cepat mendeteksi objek dengan komputasi yang rendah pada perangkat *Edge*. Diharapkan sistem *safety* ini dapat diterapkan untuk mendeteksi *person* yang memasuki area *hazard* pada lingkungan industri agar terciptanya keselamatan dan kesehatan kerja.

REFERENSI

[1] L. Ma and C. Wang, “Safety Issues in Human-Machine Collaboration and Possible Countermeasures BT - Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Anthropometry, Human Behavior, and Communication,” 2022, pp. 263–277.

[2] C. M. P. L. I. M. R. A. Simanjuntak, “Analisis Peluang Penggunaan Metode Pemindai Laser 3 Dimensi untuk Penjaminan Mutu di Era Industri 4.0 pada Proyek Konstruksi di Indonesia,” *Pros. Semin. Nas. Tek. Sipil UMS*, no. 2020: Prosiding Seminar Nasional Teknik Sipil UMS, pp. 368–374, 2020.

[3] M. Safeea and P. Neto, “Minimum distance calculation using laser scanner and IMUs for safe human-robot interaction,” *Robot. Comput. Integr. Manuf.*, vol. 58, pp. 33–42, 2019, doi: https://doi.org/10.1016/j.rcim.2019.01.008.

[4] G. Vignali *et al.*, “Performance evaluation and cost analysis of a 2D laser scanner to enhance the operator’s safety,” in *2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 2019, pp. 1–6. doi: 10.1109/ICE.2019.8792567.

[5] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” in *Computer Vision -- ECCV 2016*, 2016, pp. 21–37.

[6] A. Kumar, Z. J. Zhang, and H. Lyu, “Object detection in real time based on improved single shot multi-box detector algorithm,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2020, no. 1, p. 204, 2020, doi: 10.1186/s13638-020-01826-x.

[7] Q. Shuai and X. Wu, “Object detection system based on SSD algorithm,” in *2020 International Conference on Culture-oriented Science & Technology (ICCST)*, 2020, pp. 141–144. doi: 10.1109/ICCST50977.2020.00033.

[8] X. Deng and S. Li, “An Improved SSD Object Detection Algorithm Based on Attention Mechanism and Feature Fusion,” *J. Phys. Conf. Ser.*, vol. 2450, p. 12088, 2023, doi: 10.1088/1742-6596/2450/1/012088.

[9] S. Paul, “Optimizing MobileDet for Mobile Deployments,” *sajak.dev*, 2020. https://sayak.dev/mobiledet-optimization/ (accessed Oct. 06, 2023).

[10] Y. Xiong *et al.*, “MobileDets: Searching for Object Detection Architectures for Mobile Accelerators,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3824–3833. doi: 10.1109/CVPR46437.2021.00382.

[11] K. Seshadri, B. Akin, J. Laudon, R. Narayanaswami, and A. Yazdanbakhsh, “An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks,” in *2022 IEEE International Symposium on Workload Characterization (IISWC)*, 2022, pp. 79–91. doi: 10.1109/IISWC55918.2022.00017.

[12] A. M. Kist, “Deep Learning on Edge TPUs,” *CoRR*, vol. abs/2108.1, 2021.

[13] R. Galliera and N. Suri, “Object Detection at the Edge: Off-the-shelf Deep Learning Capable Devices and Accelerators,” *Procedia Comput. Sci.*, vol. 205, pp. 239–248, 2022, doi: https://doi.org/10.1016/j.procs.2022.09.025.

[14] B. Kovács, A. D. Henriksen, J. D. Stets, and L. Nalpantidis, “Object Detection on TPU Accelerated Embedded Devices,” in *Computer Vision Systems*, 2021, pp. 82–92.

[15] B. Pang, E. Nijkamp, and Y. N. Wu, “Deep Learning With TensorFlow: A Review,” *J. Educ. Behav. Stat.*, vol. 45, pp. 227–248, 2020.

[16] M. Abadi, M. Isard, and D. G. Murray, “A Computational Model for TensorFlow: An Introduction,” in *Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2017, pp. 1–7. doi: 10.1145/3088525.3088527.

[17] D. Smilkov *et al.*, “TensorFlow.js: Machine Learning for the Web and Beyond,” *ArXiv*, vol. abs/1901.0, 2019.

[18] A. Cardoso, J. Leitão, and C. Teixeira, “Using the Jupyter Notebook as a Tool to Support the Teaching and Learning Processes in Engineering Courses,” in *The Challenges of the Digital Transformation in Education*, 2019, pp. 227–236.

[19] N. Silaparasetty, “Machine Learning Concepts with Python and the Jupyter Notebook Environment: Using Tensorflow 2.0,” *Mach. Learn. Concepts with Python Jupyter Noteb. Environ.*, 2020.

[20] B. Jabir, F. Noureddine, and K. Rahmani, “Accuracy and Efficiency Comparison of Object Detection Open-Source Models,” *Int. J. Online Biomed. Eng.*, vol. 17, p. 165, 2021, doi: 10.3991/ijoe.v17i05.21833.

[21] D. Vogelsang and B. Erickson, “Magician’s Corner: 6. TensorFlow and TensorBoard,” *Radiol. Artif. Intell.*, vol. 2, p. e200012, 2020, doi: 10.1148/ryai.2020200012.

[22] N. Vu, “Deploy your own SSDLite Mobiledet object detector on Google Coral’s EdgeTPU using Tensorflow’s Object Detection API,” *namburger.medium.com*, 2020. https://namburger.medium.com/deploy-your-own-ssdlite-mobiledet-object-detector-on-google-corals-edgetpu-using-tensorflow-s-f41f1e3360c8 (accessed Oct. 06, 2023).

[23] R. Padilla, S. L. Netto, and E. A. B. da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242. doi: 10.1109/IWSSIP48289.2020.9145130.

[24] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, 2022, doi: 10.1109/TNNLS.2021.3084827.

[25] B. Kovács, A. D. Henriksen, J. D. Stets, and L. Nalpantidis, “Object Detection on TPU Accelerated Embedded Devices BT - Computer Vision Systems,” 2021, pp. 82–92.

LAMPIRAN 1

FORMULIR LOOGBOOK BIMBINGAN DAN PENGAJUAN SEMINAR PROPOSAL/SIDANG TUGAS AKHIR*

Nama : Abdillah Fikri
NIM : 4222001029
Pembimbing : Eko Rudiawan Jamzuri S.ST, M.Sc
Judul : Implementasi Algoritma SSD MobileDets untuk Mendeteksi Aktivitas Operator Memasuki Area Hazard di Sekitar Mesin

No	Hari/Tgl	Rincian Kegiatan	TTD Pembimbing
1	05/09-2023	Menentukan judul topik tugas akhir	
2	14/09-2023	Mencari referensi topik dan risat	
3	21/09-2023	Melakukan bimbingan berkala	
4	25/09-2023	Mengumpulkan data Image untuk pelatihan	
5	02/10-2023	Melakukan pelatihan model	
6	05/10-2023	Menguji modal dengan Image test	
7	12/10-2023	Pertemuan terkait perkembangan risat	
8	20/10-2023	Pengembangan Progam pendeteksian	
9	02/11-2023	Uji coba progam menggunakan Raspberry Pi 4	
10	10/11-2023	Melakukan pengujian sistem	
11	16/11-2023	Melakukan bimbingan dan perbaikan sistem	
12	23/11-2023	Finalisasi sistem dan pengambilan data	
13	30/11-2023	Pembahasan evaluasi dan kinerja	
14	07/12-2023	Pembuatan buku tugas akhir dan validasi dosen	

Berdasarkan hasil bimbingan yang telah dilaksanakan selama 4 bulan dan telah disetujui oleh dosen pembimbing, maka dengan ini saya mengajukan diri sebagai peserta Seminar Proposal /Sidang Tugas Akhir*.

Batam, 20 Desember 2023
Peserta

NIM: 4222001029