Pengembangan Aplikasi untuk Mendeteksi Penggunaan Masker Berbasis TinyML di Raspberry Pi

Steven Reynandi Owen[#1], Yoyok Gamaliel[#2], Herry Imanta Sitepu[#3]

[#]*Program Studi Teknik Elektro, Institut Teknologi Harapan Bangsa*
Jl. Dipatiukur no 80-84, Bandung, Jawa Barat, Indonesia

[1]1318002srowen@gmail.com

[2]yoyok@ithb.ac.id

[3]hisitepu@gmail.com

Abstract*— Currently, the rapid spread of the COVID-19 virus is hitting global health. According to the World Health Organization (WHO), one way to prevent the spread of the corona virus is to wear a medical mask. Monitoring the use of masks in public places can be a challenge because manual monitoring can lead to errors and the risk of COVID-19 transmission. Based on these problems, there is an alternative solution in the form of using artificial intelligence to detect the use of masks and body temperature. However, the model used in the alternative consumes a large amount of resources. This paper proposes an application that uses a TinyML-based machine learning model to detect face masks. The result is a cost-effective and reliable method of using AI and sensors to monitor areas where masks are required. Using TensorFlow Lite, models can be modified and run on Internet of Things (IoT) devices that have limited power. The modified model will be run on the Raspberry Pi to detect the use of a face mask and if someone is not wearing a face mask, the system will turn on an alarm in the form of a red LED light and a buzzer that sounds three times. The proposed model can be used at the entrance of shopping malls, hotels, airport entrances, etc.

***Keywords*— COVID-19, Face mask detection, Machine learning, Raspberry Pi, TinyML, TensorFlow**

***Abstrak*— Saat ini, penyebaran cepat virus COVID-19 sedang melanda kesehatan global. Menurut Organisasi Kesehatan Dunia (WHO), salah satu cara mencegah penyebaran virus corona adalah memakai masker medis. Memantau penggunaan masker di tempat umum dapat menjadi tantangan karena pemantauan secara manual bisa saja terdapat error dan risiko penularan COVID-19. Berdasarkan permasalahan tersebut, terdapat solusi alternatif berupa penggunaan kecerdasan buatan untuk mendeteksi penggunaan masker dan suhu tubuh. Namun, model yang digunakan dalam alternatif tersebut menghabiskan sumber daya yang besar. Makalah ini mengusulkan aplikasi yang menggunakan model pembelajaran mesin (*machine learning*) berbasis TinyML untuk mendeteksi masker wajah. Hasilnya, metode penggunaan AI dan sensor yang hemat biaya dan andal untuk memonitor kawasan yang wajib menggunakan masker. Dengan menggunakan TensorFlow Lite, model dapat diubah dan dijalankan pada perangkat Internet of Things (IoT) yang memiliki daya terbatas. Model yang telah diubah akan dijalankan pada Raspberry Pi untuk mendeteksi penggunaan masker wajah dan bila seseorang tidak menggunakan masker wajah, maka sistem akan menyalakan alarm berupa lampu LED berwarna merah dan buzzer yang berbunyi sebanyak tiga kali. Model yang diusulkan dapat digunakan di pintu masuk pusat perbelanjaan, hotel, pintu masuk bandara, dan lain-lain.**

***Kata Kunci*— Covid-19, Mendeteksi Masker, Machine Learning, TinyML, Tensorflow Lite. Raspberry Pi.**

I. PENDAHULUAN

Pandemi Covid-19 telah melanda Indonesia sejak Maret 2020, pemerintah Indonesia telah mengupayakan berbagai hal dalam menanggulangi pandemi Covid-19, salah satunya dengan menerapkan protokol kesehatan berupa memakai masker [1]. Hal ini dilakukan oleh pemerintah agar penularan Covid-19 dapat dikendalikan. Meskipun protokol kesehatan tersebut diterapkan, masih terdapat banyak warga yang tidak mematuhi protokol kesehatan tersebut dengan tidak memakai masker [2]. Mendeteksi penggunaan masker pengunjung di pintu masuk menjadi hal penting dan langkah ini sebagai pencegahan infeksi virus Covid-19. Oleh karena itu, sebuah detektor masker wajah otomatis harus dipasang di pintu masuk kawasan wajib masker seperti bandara, supermarket, dan semua lembaga layanan publik [3].

Beberapa hal dapat diterapkan untuk mengatasi masalah tersebut, salah satu solusi yang ditawarkan adalah mengembangkan sebuah aplikasi untuk mendeteksi masker menggunakan model *machine learning* yang menerapkan *image processing* untuk memproses data visual dan mendeteksi pengunjung yang tidak menggunakan masker [3]. Terdapat berbagai riset dalam mendeteksi penggunaan masker dengan *image processing* seperti sistem pendeteksi masker sebagai alat pengawasan di kawasan publik oleh Arjya Das dan kawan-kawan [4]. Selain itu terdapat sistem kontrol pintu masuk untuk mendeteksi suhu tubuh dan penggunaan masker yang dikembangkan oleh B. Varshini dan kawan-kawan [5]. Berbagai perusahaan yang mengembangkan solusi dalam hal mendeteksi penggunaan masker diantaranya adalah LeewayHertz dan SightCorp dengan solusi berupa aplikasi yang menggunakan *machine learning* (ML) untuk mendeteksi penggunaan masker secara langsung (*real-time*) dari kamera seperti CCTV [6] [7].

Namun berdasarkan solusi di atas, machine learning yang menerapkan *image processing* memiliki komputasi yang intensif dan performa yang tinggi [3]. Hal ini dapat menjadi masalah bila solusi membutuhkan biaya tinggi dan apabila ingin diterapkan pada *edge device* yang memiliki sumber daya dan spesifikasi yang terbatas. Istilah *edge device* adalah peralatan dengan *resource* terbatas yang berfungsi untuk mendapatkan dan mengirimkan data antara jaringan lokal dan cloud [8]. Bila solusi dapat dijalankan pada *edge device*, tentunya hal ini dapat mendukung efektivitas pengawasan penggunaan masker pada pintu masuk kawasan wajib masker. Selain itu, faktor biaya juga merupakan faktor, karena perangkat yang lebih murah akan menghasilkan aksesibilitas yang lebih tinggi [9].

TinyML adalah konsep dimana arsitektur dan pendekatan *machine learning* yang sama digunakan, tetapi pada perangkat yang lebih kecil yang mampu melakukan fungsi yang berbeda, dari menjawab perintah *audio* hingga menjalankan tindakan tertentu melalui interaksi [10]. Tentunya dengan perangkat yang kecil ini, TinyML memiliki fleksibilitas dan biaya perawatan yang rendah. Oleh karena itu, TinyML memungkinkan model *machine learning* untuk berjalan pada jaringan latensi rendah, konsumsi daya rendah, dan inferensi model *bandwidth* rendah di *edge device* [11].

Untuk mencapai hal di atas, dibutuhkan suatu alat (tools) untuk mengkonversi model *machine learning* yang semula berukuran besar menjadi kecil. Tensorflow Lite merupakan alat yang populer digunakan untuk menjalankan model *machine learning* di dalam *embedded system* [12]. Dengan Tensorflow Lite, model *machine learning* dapat dikelompokkan dan diubah sehingga model dapat dijalankan pada *embedded system* seperti Arduino [13].

Dalam penelitian ini, *edge device* yang akan dipakai untuk implementasi TinyML adalah Raspberry Pi 3 model B. Raspberry Pi merupakan salah satu perangkat yang populer digunakan dalam IoT. Hal ini dikarenakan Raspberry Pi memiliki CPU yang kuat yang digabungkan dengan LAN Nirkabel dan radio Bluetooth menjadikannya kandidat ideal untuk proyek IoT, karena beberapa sensor dapat dihubungkan secara bersamaan [14]. Selain itu, Raspberry Pi memiliki konektor GPIO (General Purpose I/O) 40-pin untuk berinteraksi dengan sensor eksternal [15].

Berdasarkan dari permasalahan di atas, dibutuhkan sebuah sistem yang mengaplikasikan TinyML untuk mendeteksi masker. TinyML dibutuhkan agar sistem dapat berjalan pada edge device yang memiliki sumber daya dan spesifikasi yang terbatas seperti Raspberry Pi. Hasil akhir penelitian berupa sebuah sistem yang mengaplikasikan TinyML untuk mendeteksi masker, dapat berjalan pada Raspberry Pi, dan dapat memberikan peringatan berupa lampu LED berwarna merah dan buzzer yang berbunyi sebanyak tiga kali bila ada pengunjung yang tidak menggunakan masker.

II. METODOLOGI

Dalam implementasi dan pengembangan aplikasi untuk mendeteksi masker berbasis monitoring dan tracking pasien Covid-19 berbasis TinyML di Raspberry Pi, perlu dilakukan peninjauan pustaka sebagai dasar dari perancangan dan pengembangan sistem. Tinjauan pustaka akan dilakukan dengan mengidentifikasi masalah, menganalisis sistem yang sudah ada, sistem yang diusulkan, dan analisis kebutuhan sistem.

A. Identifikasi Masalah

Pandemi Covid-19 di Indonesia belum berakhir, protokol kesehatan dalam hal memakai masker masih diterapkan di ruang publik meskipun pemerintah telah melonggarkan protokol kesehatan tersebut [16]. Pengawasan penggunaan masker di kawasan wajib masker diperlukan agar penularan Covid-19 dapat dikendalikan dengan baik. Salah satu solusi yang telah ditempuh adalah pengembangan berbagai jenis aplikasi *machine* learning pendeteksi masker [3].

Namun, model *machine* learning yang mengolah data gambar memiliki data membutuhkan banyak memori dalam melatih dan menjalankan *machine* learning. Hal ini dikarenakan dalam pengolahan gambar di dalam model *machine* learning membutuhkan memori untuk menyimpan data input, parameter bobot dan aktivasi sebagai input yang merambat melalui jaringan model *machine* learning tersebut [17]. Sehingga dapat dikatakan model *machine* learning untuk *image processing* memiliki komputasi yang intensif dan performa yang tinggi [3]. Tentunya dengan memori yang terpakai besar dapat menjadi hambatan dalam menjalankan model di *edge device*.

Dalam menjalankan model pada *edge devices*, terdapat sebuah kendala utama yaitu apakah model tersebut dapat berjalan dengan *resource* (memori, CPU, dan lain-lain) yang terbatas [18]. Kendala tersebut dapat mempengaruhi performa dalam hal mendeteksi masker, performa yang lambat dapat mengakibatkan penumpukan pengunjung dikarenakan proses pemeriksaan yang lambat [19]. Selain itu, penggunaan *resource* yang besar juga memakan biaya operasi yang besar [9].

B. Riset Terkait

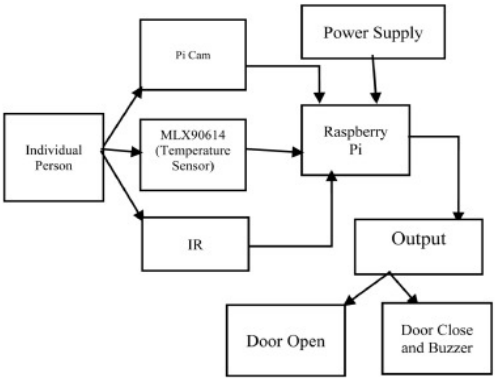
Dalam melakukan penelitian terdapat beberapa riset yang terkait dengan aplikasi *machine learning* untuk mendeteksi masker. Riset pertama adalah Mohammad Farid Naufal, Selvia Ferdiana Kusuma, Zefanya Ardya Prayuska, dan kawan-kawan melakukan riset pada tahun 2021 dengan judul *Comparative Analysis of Image Classification Algorithms for Face Mask Detection*. Mereka memiliki objektif untuk membandingkan algoritma klasifikasi *machine learning* klasik sperti *k-nearest neighbor* (KNN), *support vector machine* (SVM), dan *convolutional neural network* (CNN) untuk deteksi masker wajah. Hasil dari riset mereka adalah CNN memiliki rata-rata kinerja terbaik dengan akurasi 0,9683 dan waktu eksekusi rata-rata 2.507,802 detik untuk mengklasifikasikan 3.725 wajah dengan masker dan 3.828 wajah tanpa gambar masker [20].

Riset selanjutnya adalah B. Varshini, H.R.Yogesh, Syed Danish Pasha, Maaz Suhail, V. Madhumitha, Archana Sasi melakukan riset pada tahun 2021 mengenai *IoT-Enabled Smart Doors for Monitoring Body Temperature and Face*

Mask Detection. Objektif dari riset mereka adalah membuat sebuah alat IoT untuk mendeteksi masker wajah dan suhu tubuh dengan Raspberry Pi yang dapat digunakan untuk pintu masuk mall, hotel, dan lain-lain. Hasil akhir yang didapat dari riset mereka berupa sebuah alat IoT yang mendeteksi masker dan suhu tubuh dengan tingkat akurasi diatas 90%. [5].

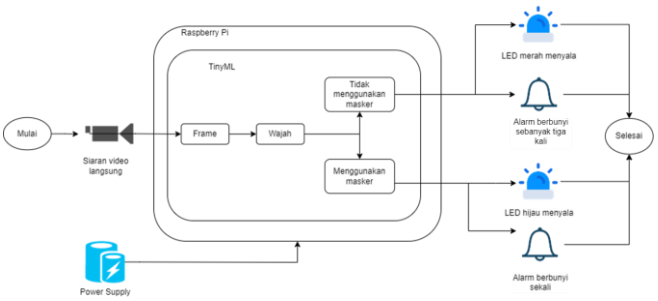
C. *Sistem yang Sudah Ada dan Sistem yang Diusulkan*

Dapat dilihat dalam Gambar 1, sebuah sistem smart door diusulkan oleh B. Varsini dan kawan-kawan. Sistem ini akan mendeteksi penggunaan masker dan suhu tubuh pengunjung. Dalam sistem tersebut, kamera akan mendapat input dari gambar ataupun video. Model yang telah dikembangkan akan mendeteksi setiap wajah dan akan melakukan deteksi apakah suhu tubuh normal dan wajah bermasker atau tidak., Output dari sistem berupa pintu yang ditutup atau terbuka berdasarkan hasil prediksi deteksi masker dan suhu tubuh pengunjung



Gambar 1 Sistem yang Sudah Ada

Adapun sistem yang diusulkan (dapat dilihat pada Gambar 2) adalah sebagai berikut. Kamera akan melakukan video stream secara langsung dengan output berupa *frame*. *Frame* akan dianalisa oleh sebuah detektor wajah yang menggunakan *Deep Neural Networ*k (DNN) dari OpenCV versi 3.3. Detektor wajah ini berupa model Caffe yang sudah dilatih (pre-trained) yang menggunakan *Single Shot-Multibox Detector* (SSD) dan ResNet-10. OpenCV akan mendeteksi wajah menggunakan model Caffe. Setelah wajah dideteksi, Tensorflow Lite akan mengidentifikasi apakah individu menggunakan masker wajah atau tidak. Bila ada individu yang tidak menggunakan masker, sistem akan menyalakan lampu peringatan berupa LED berwarna merah dan akan membunyikan alarm peringatan sebanyak tiga kali.



Gambar 2 Sistem yang Diusulkan

D. *Analisis Kebutuhan*

Dalam bagian analisis kebutuhan, kebutuhan sistem yang disusun terbagi menjadi menjadi dua bagian, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional merupakan kebutuhan yang berisi proses atau layanan apa saja yang nantinya harus disediakan oleh sistem. Sedangkan kebutuhan non fungsional merupakan spesifikasi yang dibutuhkan oleh sistem.

Adapun kebutuhan-kebutuhan fungsional sistem dapat dilihat dalam Tabel I di bawah ini:

TABEL I
TABEL KEBUTUHAN NON FUNGSIONAL

No ID	Fungsi
F01	Pengguna akan mendapatkan peringatan ketika ada pengunjung yang tidak menggunakan masker berupa lampu LED berwarna merah yang menyala.
F02	Pengguna akan mendapatkan peringatan ketika ada pengunjung yang tidak menggunakan masker berupa *buzzer* yang berbunyi sebanyak tiga kali.
F03	Pengguna akan mendapatkan konfirmasi ketika ada pengunjung yang menggunakan masker berupa lampu LED berwarna hijau yang menyala
F04	Pengguna akan mendapatkan konfirmasi ketika ada pengunjung yang menggunakan masker berupa *buzzer* yang berbunyi satu kali
F05	Sistem dapat menyalakan LED merah ketika ada pengunjung yang tidak menggunakan masker
F06	Sistem dapat membunyikan buzzer sebanyak tiga kali ketika ada pengunjung yang tidak menggunakan masker
F07	Sistem dapat menyalakan LED hijau ketika ada pengunjung yang menggunakan masker
F08	Sistem dapat membunyikan buzzer satu kali ketika ada pengunjung yang menggunakan masker
F09	Pengunjung mampu melihat hasil prediksi mendeteksi masker dari model di layar tampilan
F10	Pengguna mampu melihat hasil prediksi mendeteksi masker di layar tampilan

Adapun kebutuhan-kebutuhan fungsional sistem dapat dilihat dalam Tabel II di bawah ini:

TABEL II
TABEL KEBUTUHAN NON FUNGSIONAL

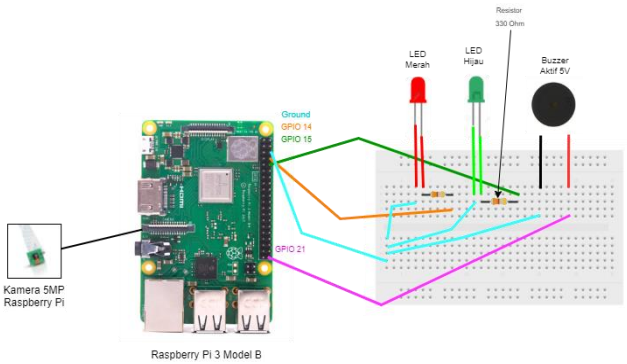
No ID	Fungsi
F01	Sistem dapat mendeteksi wajah dengan secara langsung (video stream).
F02	Sistem dapat mengaplikasikan TinyML untuk mendeteksi masker
F03	Sistem dapat berjalan di Raspberry Pi dengan *resource* terbatas

E. Perancangan dan Implementasi Sistem

Perancangan sistem dibagi menjadi 3 bagian yaitu rancangan sistem perangkat keras, perancangan sistem perangkat lunak, dan integrasi sistem.

1) Perancangan Perangkat Keras

Dalam penelitian ini, sistem menggunakan perangkat keras sebagai *input* dan *output* dari sistem, *output* sistem berupa lampu LED dan *buzzer* dapat dilihat oleh pengunjung. Skematik dari keseluruhan perancangan perangkat keras dapat dilihat pada Gambar 3, kamera 5MP Raspberry Pi terhubung dengan port kamera yang terdapat pada Raspberry Pi 3B. GPIO 14 (Tx) dihubungkan dengan pin positif LED merah dan diberi resistor sebesar 330 Ohm untuk menyalakan LED merah dan GPIO 15 (Rx) dihubungkan dengan pin positif LED hijau dan diberi resistor sebesar 330 Ohm untuk menyalakan LED hijau. Sedangkan, GPIO 21 (SCLK) dihubungkan dengan pin positif buzzer. Terakhir, seluruh sirkuit dihubungkan dengan ground Raspberry Pi.



Gambar 3 Skematik Keseluruhan Perangkat

2) Perancangan Perangkat Lunak

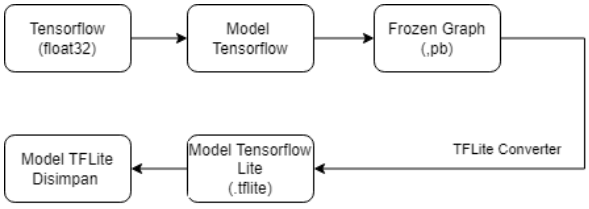
Perangkat lunak digunakan oleh sistem untuk memproses input sistem yang didapatkan dari kamera dan mengeluarkan output sistem berupa LED dan buzzer. Dalam penelitian ini, perangkat lunak akan dijalankan di dalam Raspberry Pi 3B. Adapun tahap-tahap dalam merancang perangkat lunak terbagi menjadi dua tahap, yaitu pelatihan model Tensorflow dan konversi model Tensorflow menjadi model Tensorflow Lite.

Dalam melatih model Tensorflow, dibutuhkan dataset sebagai acuan bagi model dalam membedakan pengunjung yang menggunakan masker atau tidak. Dataset terdiri dari dua kategori yaitu bermasker dan tidak bermasker. Jumlah dataset total adalah 3.911 files (155 MB) dengan jumlah dataset bermasker adalah 1.968 files (114 MB) dan jumlah dataset yang tidak bermasker adalah 1.943 (41 MB). Gambar diubah dahulu (preprocess) dengan Keras sebelum diubah menjadi data *array*. Data *array* diubah menjadi Numpy *array* dengan tipe 'float32'. Numpy *array* kemudian dipisah menjadi *data training* dan *data test* dengan rasio standar sebesar 75% dan 25%. Setelah dipisah, gambar dibangun menjadi *training image generator* untuk augmentasi data dengan parameter sebagai berikut: rotasi sebesar 45 derajat, *zoom* sebesar 15%, pemindahan ke atas dan samping sebesar 20%, pengubahan sudut gambar sebesar 15%, gambar dibalik secara horizontal, dan *fill mode* adalah *nearest*

Dalam pembuatan model, *baseModel* dari MobileNetV2 dimuat dahulu. MobileNetV2 adalah aplikasi Keras untuk memproses gambar dengan algoritma convolutional neural network (CNN) dan kedalaman 53 layer. *headModel* (layer tambahan) dibangun berdasarkan *baseModel* untuk menyesuaikan model yang diinginkan, lalu *headModel* ditambahkan di atas baseModel. Selanjutnya, seluruh layer di *baseModel* dibekukan.

Model dioptimalisasi dengan Adam dengan *learning rate* sebesar 0.0001 dan *batch size* sebesar 32. Setelah optimisasi, model dilatih berdasarkan model yang telah disusun. Setelah model dilatih, model akan melakukan prediksi berdasarkan data *test* yang telah dipisahkan sebelumnya.. Bila model sudah melakukan prediksi, model akan menuliskan hasil akhir dari pelatihan dan membuat grafik berdasarkan hasil prediksi. Terakhir, model Tensorflow ditulis pada *disk*

Tahap selanjutnya adalah konversi model Tensorflow menjadi model Tensorflow Lite, tahap ini dapat dilihat dalam Gambar 4. Model Tensorflow yang tersimpan akan dimuat untuk diubah dengan TFLite Converter. Setelah model dimuat, model akan diubah menjadi model Tensorflow Lite tanpa melakukan kuantisasi. Pengubahan tanpa kuantisasi ini dilakukan untuk menjaga keakuratan dari model Tensorflow yang telah dilatih. Setelah diubah, model disimpan ke dalam *disk*



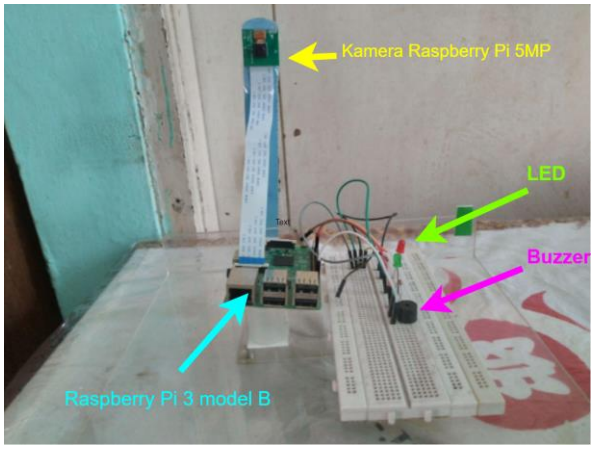
Gambar 4 Diagram Blok Konversi Mode Tensorflow

3) Integrasi Sistem

Setelah melakukan perancangan perangkat keras dan lunak, tahap selanjutnya adalah melakukan integrasi sistem. Pertama-tama, kamera dijalankan dan siaran video langsung akan dimulai.Sebelum proses melakukan looping, model Caffe untuk deteksi wajah dan model TFLite akan dimuat dari disk. Proses masuk ke dalam looping, kamera akan menangkap *frame* dan melakukan pengaturan ukuran *frame* tersebut sebelum dimasukan ke dalam fungsi. Di dalam fungsi, dimensi *frame* akan diambil dan blob akan disusun dari dimensi *frame* yang terambil.

Blob (*Binary large object*) adalah sebuah group dari pixel (dalam sebuah *image*) yang saling terhubung dengan properti yang sama. Blob akan dioperkan melalui jaringan dan deteksi wajah didapatkan. Kemudian, mengambil kemungkinan berdasarkan deteksi wajah yang didapatkan. Deteksi wajah yang lemah akan disaring. Lalu, melakukan perhitungan untuk menentukan lokasi ROI (Region of Interest) wajah. Setelah itu, sistem akan menaruh box di sekitar hasil wajah yang terdeteksi. Hasil wajah diekstrak lalu diubah dari BGR menjadi RGB channel. Wajah yang terdeteksi kemudian diubah ukurannya lalu pra proses hasil wajah sebelum dikirim ke TFLite. Wajah yang telah diproses kemudian dideteksi oleh TFLite untuk mendapatkan hasil prediksi bermasker atau tidak. Hasil prediksi kemudian didapatkan dan proses keluar dari fungsi.

Setelah keluar dari fungsi, hasil prediksi kemudian dibandingkan. Bila hasil prediksi menyatakan wajah tidak menggunakan masker, maka lampu LED berwarna merah akan menyala dan buzzer akan berbunyi sebanyak 3 kali. Sebaliknya, bila pengunjung menggunakan masker, maka lampu LED berwarna hijau akan menyala dan buzzer akan berbunyi satu kali selama 2 detik, lalu sistem akan menunggu selama 1 detik. Gambar 5 menunjukan prototipe sistem yang dikembangankan.



Gambar 5 Prototipe Perangkat yang Telah Digabung

III. HASIL DAN PEMBAHASAN

Dalam menguji perangkat lunak untuk mendeteksi penggunaan masker dalam penelitian ini, peneliti menggunakan teknik pengujian *Black Box Testing* untuk mengamati hasil *input* dan *output* dari perangkat lunak yang telah dikembangkan. Adapun input dalam sistem ini berupa *frame* yang diambil dari siaran video langsung. Untuk output aplikasi berupa lampu LED yang menyala dan buzzer yang berbunyi sesuai dengan hasil prediksi Tensorflow Lite.

A. Tujuan dan Skenario Pengujian

Pengujian terhadap sistem pemeriksaan penggunaan masker ini dilakukan untuk memastikan tujuan penelitian tercapai. Adapun tujuan penelitian ini adalah untuk mengembangkan suatu sistem yang mendeteksi penggunaan masker berbasis TinyML yang dapat berjalan di Raspberry Pi. Dalam mencapai tujuan pengujian tersebut, skenario pengujian merupakan suatu hal yang penting agar memudahkan tahapan pengujian sistem dan mendapatkan hasil yang diharapkan. Hal ini bertujuan agar sistem dapat menjalankan fungsinya dengan baik dan memiliki performa yang optimal. Terdapat dua jenis waktu yang dipakai dalam skenario pengujian, yaitu wall clock time dan CPU time. Wall clock time adalah waktu yang berlalu menurut jam internal komputer, yang seharusnya sesuai dengan waktu di dunia luar. Ini tidak ada hubungannya dengan penggunaan CPU. CPU time adalah berapa lama proses spesifik dihabiskan secara aktif untuk dieksekusi di CPU. Pengukuran waktu ini dilakukan untuk menemukan suatu sistem pemeriksa masker yang memiliki performa dan waktu terbaik dalam mendeteksi masker dan dapat berjalan di dalam Rasberry Pi Adapun persamaan dari wall clock time dan CPU time dapat dilihat dalam Persamaan 1 di bawah ini.

$$\Delta time = end\ time - start\ time \quad (1)$$

Berdasarkan penjelasan di atas, skenario pengujian dibagi menjadi 7 skenario yang dapat dilihat dalam Tabel III.

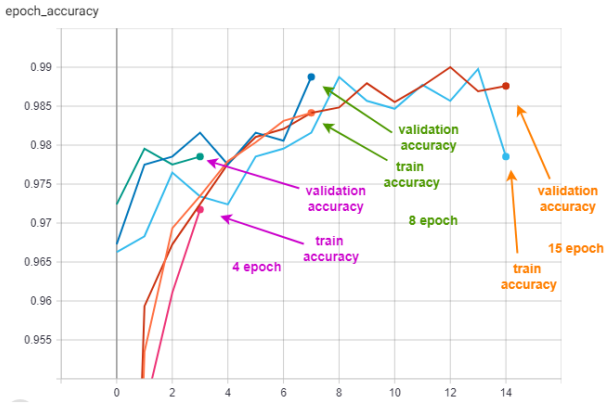
TABEL III
TABEL SKENARIO PENGUJIAN

No	Skenario Pengujian	Waktu yang diuji
1	Pengaruh jumlah epoch saat pelatihan model terhadap *training time*, *epoch accuracy*, dan *epoch loss*	*Wall clock time*
2	Pengaruh *Graphical User Interface* (GUI) terhadap performa sistem	*CPU time*
3	Performa sistem dalam menjalankan model Tensorflow dan model Tensorflow Lite	*CPU time*
4	Waktu yang dibutuhkan untuk mendapatkan hasil prediksi	*Wall clock time*
5	Pengaruh jarak kamera terhadap hasil prediksi	-
6	Hasil prediksi penggunaan masker wajah	-
7	Akurasi hasil prediksi terhadap lima jenis masker wajah yang berbeda	-

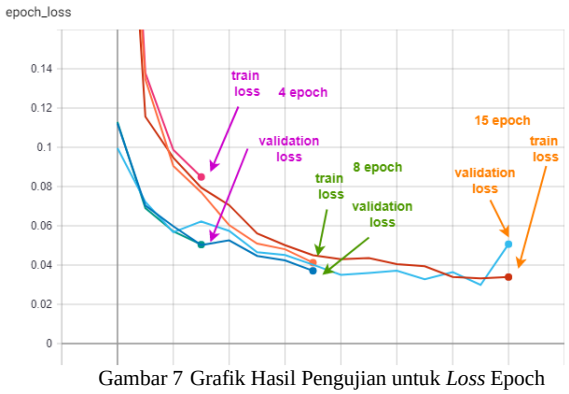
B. *Hasil dan Analisis Pengujian*

Setiap skenario pengujian dilakukan dengan melakukan iterasi sebanyak 20 kali. Hasil pengujian akan berupa rata-rata dari data yang telah diperoleh. Dalam pengujian, wajah yang dideteksi berjarak 30 cm dari kamera kecuali pada skenario 5 yang menguji pengaruh jarak kamera.

Pengujian pertama adalah pengaruh jumlah epoch saat pelatihan model *terhadap training time, epoch accuracy,* dan *epoch loss.* Pengujian ini bertujuan untuk menemukan model yang optimal, yakni model yang memiliki training time yang lebih cepat, penurunan epoch accuracy yang rendah, dan peningkatan epoch loss yang rendah. Hal ini dilakukan untuk menghemat waktu pelatihan dengan mengorbankan akurasi sesedikit mungkin. Model yang diuji adalah model Tensorflor saat model dilatih. Pengukuran waktu pelatihan meliputi saat: model dasar MobileNetV2 dimuat, saat penyusunan kepala model dari model dasar, saat kedua model digabungkan, saat seluruh layer di model dasar dibekukan, saat optimisasi model dengan Adam, dan saat model dilatih. Grafik hasil pengujian *epoch accuracy* dan *epoch loss* dapat dilihat masing-masing dalam Gambar 6 dan Gambar 7.



Gambar 6 Grafik Hasil Pengujian untuk Akurasi Epoch



Gambar 7 Grafik Hasil Pengujian untuk *Loss* Epoch

Pada hasil akhir pengujian pertama, model 8 epoch dianggap optimal karena penurunan akurasi train sebesar 0.81% dan peningkatan train loss sebesar 21.53% namun memiliki waktu training lebih rendah sebesar 47.34% dibandingkan model 15 epoch. Sedangkan model 4 epoch tidak dianggap optimal karena memiliki penurunan train akurasi sebesar 2.02% dan peningkatan train loss sebesar 184.36% meskipun memiliki penurunan waktu training sebesar 73.17% terhadap model 15 epoch. Jadi berdasarkan hasil pengujian pertama, model 8 epoch akan dipakai dalam pengujian-pengujian selanjutnya.

Pengujian kedua adalah pengaruh GUI terhadap performa sistem. Pengujian ini bertujuan untuk mengetahui performa sistem yang lebih ketika berjalan dengan dan tanpa GUI. Model yang terhubung tanpa GUI menggunakan kabel Ethernet sebagai penyambung dan model yang menggunakan GUI terhubung langsung dengan monitor menggunakan kabel VGA. Model yang dianggap optimal adalah model yang menghabiskan waktu lebih rendah dibandingkan dengan yang lain. Seluruh model dijalankan dalam keadaan *idle* atau tanpa mendeteksi wajah. Dalam mengumpulkan data, hasil data waktu pertama setelah menjalankan sistem dapat diabaikan karena hasil data waktu pertama dapat bernilai besar dan mempengaruhi hasil akhir data yang lain.

Hasil pengujian yang diperoleh memiliki satuan milidetik. Hasil pengujian kedua dengan model Tensorflow yang tidak menggunakan GUI menghabiskan waktu lebih rendah sebesar 6,44% dan model Tensorflow Lite menghabiskan waktu lebih rendah sebesar 6,07% dibandingkan dengan model masing-masing yang menggunakan GUI. Oleh karena itu, sistem akan dijalankan tanpa GUI dalam pengujian-pengujian selanjutnya untuk menambah performa sistem

Pengujian ketiga bertujuan untuk membandingkan performa sistem ketika menjalankan model Tensorflow Lite (TFLite) dan model Tensorflow. Model yang dianggap optimal adalah model yang menghabiskan waktu lebih rendah dibandingkan model lain. Dalam mengumpulkan data, hasil data waktu pertama setelah menjalankan sistem dapat diabaikan karena hasil data waktu pertama dapat bernilai besar dan mempengaruhi hasil akhir data yang lain. Data hasil pengujian lalu dimasukan ke dalam fungsi hdrhistogram sehingga didapatkan distribusi persentil, rata-rata, nilai maksimal, dan simpangan baku dengan satuan milidetik. Adapun hasil pengujian performa sistem dapat dilihat dalam Tabel IV.

TABEL IV
TABEL HASIL PENGUJIAN PERFORMA SISTEM TIAP MODEL (MILIDETIK)

No	Model	Kondisi	Rata-rata	Maksimal	Simpangan Baku
1	Tensorflow	*Idle*	6.093	6.299	
2	Tensorflow Lite	*Idle*	5.083	5.171	
3	Tensorflow	Wajah terdeteksi	7.292	8.047	
4	Tensorflow Lite	Wajah terdeteksi	6.104	6.251	

Hasil pengujian ketiga dalam Tabel IV memberikan hasil berupa model Tensorflow Lite memiliki rata-rata waktu lebih rendah masing-masing sebesar 16.57% dalam posisi *idle* dan 16.29% ketika mendeteksi wajah dibandingkan dengan model Tensorflow.

Selanjutnya, pengujian keempat bertujuan untuk mengetahui waktu yang dibutuhkan oleh seseorang untuk mendapatkan hasil prediksi pemeriksaan masker wajah oleh model Tensorflow Lite. Dalam mengumpulkan data, hasil data waktu pertama setelah menjalankan sistem dapat diabaikan karena hasil data waktu pertama dapat bernilai besar dan mempengaruhi hasil akhir data yang lain. Hasil akhir waktu berupa rata-rata dari data waktu yang didapatkan saat melakukan uji coba. Pengujian dilakukan dengan sistem mendeteksi penggunaan masker dan aplikasi tidak mendeteksi penggunaan masker. Pengukuran waktu meliputi dari saat *frame* dibaca oleh imutils, saat pengubahan ukuran (resize) *frame*, dan saat mendapatkan hasil prediksi dari detektor muka dan TFLite. Hasil akhir pengujian keempat adalah model Tensorflow Lite memiliki performa lebih cepat sebesar 17.21% dalam mendeteksi wajah tidak bermasker dan 16.53% dalam mendeteksi wajah bermasker dibandingkan dengan model Tensorflow.

Pengujian kelima bertujuan untuk mengetahui akurasi sistem dalam mendeteksi wajah dengan jarak yang berbeda-beda. Sistem diuji dengan wajah yang terdeteksi tidak menggunakan masker. Sistem yang optimal adalah sistem yang dapat mendeteksi wajah minimal sebanyak 15 dari 20 kali pengujian yang dilakukan. Model yang diuji adalah model Tensorflow Lite dengan 8 epoch dan tanpa GUI. Pada hasil akhir pengujian, sistem dapat mendeteksi masker dari jarak 50 cm sampai 150 cm dari kamera. Terdapat penurunan akurasi sistem bila wajah yang terdeteksi memiliki jarak lebih dari 150 cm. Oleh karena itu, kamera harus diletakan pada jarak kurang dari 150 cm agar memiliki keakuratan yang optimal dalam mendeteksi wajah.

Pengujian keenam bertujuan untuk mengetahui akurasi sistem dalam mendeteksi masker wajah dengan berbagai posisi. Tentunya hasil akhir yang diingini adalah sistem dapat mendeteksi masker wajah dan menyalakan alarm yang benar. Model yang diuji adalah model Tensorflow Lite dengan 8 epoch dan tanpa GUI. Terdapat empat posisi masker yang akan diuji yaitu memakai masker secara penuh (mulu dan hidung tertutup), memakai masker di bawah hidung (hidung terbuka dan mulut tertutup), memakai masker di bawah dagu (mulut dan hidung terbuka), dan tidak memakai masker (mulut dan hidung terbuka). Hasil pengujian prediksi pengujian masker dapat dilihat dalam Tabel V.

TABEL V
TABEL HASIL PENGUJIAN PREDIKSI PENGGUNAAN MASKER WAJAH

No	Posisi masker	LED	Buzzer	Rata-rata *Confidence*
1	Menutupi hidung dan mulut	Hijau	Berbunyi satu kali	98.64
2	Di bawah hidung	Hijau	Berbunyi satu kali	98.97
3	Di bawah dagu	Merah	Berbunyi tiga kali	93.47
4	Tidak memakai masker	Merah	Berbunyi tiga kali	98.81

Berdasarkan hasil akhir pengujian, yang terdapat dalam Tabel V, model Tensorflow Lite memiliki nilai rata-rata confidence di atas 90%. Namun, model tidak dapat mendeteksi masker di bawah hidung. Hal ini dikarenakan dataset yang digunakan hanya terdiri dari dua kategori, yaitu bermasker dan tidak bermasker. Oleh karena itu, model tidak dapat mendeteksi hidung dan mendeteksi mulut sebagai acuan bahwa seseorang tidak bermasker.

Pengujian ketujuh bertujuan untuk mengetahui akurasi dari model yang dipakai dalam mendeteksi jenis-jenis masker yang berbeda dan sering dijumpai sehari-hari. Wajah yang terdeteksi tidak menggunakan masker. Sistem yang optimal adalah sistem yang dapat mendeteksi wajah minimal sebanyak 15 dari 20 kali pengujian yang dilakukan. Model yang diuji adalah model Tensorflow Lite dengan 8 epoch dan tanpa GUI. Adapun masker yang diuji dapat dilihat dalam Gambar 8 adalah masker medis standar, masker KN95, masker berwarna hitam, masker kain dua warna, dan masker kain batik.

Gambar 8 Jenis-jenis Masker yang Digunakan dalam Pengujian

Pengujian ini dilakukan karena terdapat berbagai jenis masker dalam kehidupan sehari-hari. Hasil akhir pengujian ketujuh adalah model dapat mendeteksi jenis-jenis masker yang umum digunakan oleh masyarakat. Selain itu, hasil akhir dari pengujian ini menguatkan fakta pada Tabel 4.6 mengenai model yang mendeteksi mulut sebagai acuan bahwa seseorang tidak bermasker

IV. SIMPULAN DAN SARAN

Berdasarkan hasil analisis pengujian sistem yang telah dilakukan, terdapat beberapa kesimpulan dapat diambil yaitu. sistem berhasil dalam mengimplementasikan TinyML (Tensorflow Lite) dalam Raspberry Pi dan menjalankan aplikasi dengan resource terbatas. Model Tensorflow yang

dilatih sudah memiliki akurasi sebesar 97% dengan waktu training selama 1009 detik Lalu, model Tensorflow Lite menggunakan *resource* lebih efisien ketika dalam keadaan *idle* dan mendeteksi muka sebesar 16.5% dan 16.3%. Sistem memiliki performa lebih cepat sebesar 6% bila tidak menggunakan GUI dalam keadaan *idle*. Tensorflow Lite memiliki keunggulan lebih cepat dalam mendeteksi masker dan tidak mendeteksi masker masing-masing sebesar 17.2% dan 16.5% dibandingkan Tensorflow. Sistem dapat mendeteksi masker dengan jarak kurang dari 150 cm. Keakuratan model Tensorflow Lite dalam mendeteksi penggunaan masker wajah adalah sebesar 90%.

Sistem berhasil dalam memperingatkan pengguna bila ada pengunjung yang tidak menggunakan masker dengan cara menyalakan LED berwarna merah dan membunyikan alarm sebanyak tiga kali. Berdasarkan kesimpulan yang telah didapat, TinyML cocok digunakan dalam bila ingin menjalankan *machine learning* pada perangkat IoT yang memiliki *resource* terbatas seperti Raspberry Pi.

Terdapat beberapa saran yang dapat digunakan sebagai bahan pengembangan penelitian lebih lanjut, yaitu dapat dipertimbangkan untuk menggunakan Raspberry Pi dengan spesifikasi di atas dari Raspberry Pi 3 model B yang digunakan, menambah tingkat keamanan untuk mencegah individual tidak bermasker yang tidak mengindahkan peringatan dan menerobos masuk ke dalam kawasan wajib masker, dan melakukan kuantisasi dalam mengubah model Tensorflow menjadi Tensorflow Lite untuk mengurangi ukuran file dan menambah performa sistem pendeteksi masker

DAFTAR REFERENSI

[1] Kominfo. (29 Juli 2021). "Pemerintah Terus Dorong Pengendalian Laju Penyebaran Covid-19," [Daring]. Tersedia: https://www.kominfo.go.id/content/detail/36050/pemerintah-terus-dorong-pengendalian-laju-penyebaran-covid-19/0/berita [Diakses: 5 Juni 2022]

[2] Pambudhy, Agung. (13 Mei 2022). "Ngeyel Sih! Warga Tak Bermasker Dihukum Nyapu Jalan," [Daring]. Tersedia: https://news.detik.com/foto-news/d-6076515/ngeyel-sih-warga-tak-bermasker-dihukum-nyapu-jalan [Diakses: 5 Juni 2022]

[3] Said, Yahia, "Pynq-YOLO-Net: An embedded quantized convolutional neural network for face mask detection in COVID-19 pandemic era," *Int. J. Adv. Comput. Sci. Apple* 11.9, 2020.

[4] Das, Arjya, M. W. Ansari, dan R. Basak, "Covid-19 face mask detection using TensorFlow, Keras and OpenCV," *2020 IEEE 17th India Council International Conference* (INDICON), IEEE, 2020.

[5] Varshini, B., et al, "IoT-Enabled Smart Doors for Monitoring Body Temperature and Face Mask Detection," *Global Transitions Proceedings*, 2021.

[6] LeewayHertz. (2020). "Face Mask Detection System using Artificial Intelligence,". [Daring]. Tersedia: https://www.leewayhertz.com/face-mask-detection-system/ [Diakses: 10 Oktober 2021]

[7] SightCorp. (2022). "Face Mask Detection," [Daring]. Tersedia: https://sightcorp.com/face-mask-detection/ [Diakses: 11 Oktober 2021]

[8] Red Hat. (4 Maret 2021). "What is IoT Edge computing?," [Daring]. Tersedia: https://www.redhat.com/en/topics/edge-computing/iot-edge-computing-need-to-work-together [Diakses: 5 Juni 2022]

[9] Piątkowski, Dominik, dan K. Walkowiak, "TinyML-Based Concept System Used to Analyze Whether the Face Mask Is Worn Properly in Battery-Operated Conditions," *Applied Sciences* 12.1, 2022.

[10] Warden, Pete, dan D. Situnayake. TinyML. *O'Reilly Media Incorporated*, 2019.

[11] Sanchez-Iborra, R., dan Skarmeta, A. F., "Tinyml-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, 20(3), 4-18, 2020.

[12] Tensorflow. (Mei 2022). "TensorFlow Lite,". [Daring]. Tersedia: https://www.tensorflow.org/lite/guide [Diakses: 5 Juni 2022]

[13] Warden, Pete, dan D. Situnayake, *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers. O'Reilly Media*, 2019.

[14] Zhao, Cheah Wai, J. Jegatheesan, dan S. C. Loon, "Exploring iot application using raspberry pi," *International Journal of Computer Networks and Applications* 2.1, 2015.

[15] Richardson, Matt, dan S. Wallace, *Getting started with raspberry PI. O'Reilly Media, Inc*., 2012.

[16] Satuan Tugas Penanganan COVID-19. (17 Mei 2022). "Pemerintah Longgarkan Kebijakan Pemakaian Masker bagi Masyarakat," [Daring]. Tersedia: https://covid19.go.id/artikel/2022/05/17/pemerintah-longgarkan-kebijakan-pemakaian-masker-bagi-masyarakat [Diakses: 5 Juni 2022]

[17] Hanlon, Jamie. (31 Januari 2017). "WHY IS SO MUCH MEMORY NEEDED FOR DEEP NEURAL NETWORKS?," [Daring]. Tersedia: https://www.graphcore.ai/posts/why-is-so-much-memory-needed-for-deep-neural-networks [Diakses: 5 Juni 2022]

[18] Zhang, Xingzhou, Y. Wang, dan W. Shi. "{pCAMP}: Performance Comparison of Machine Learning Packages on the Edges," *USENIX workshop on hot topics in edge computing* (HotEdge 18), 2018.

[19] Vijitkunsawat, Wuttichai, dan P. Chantngarm, "Study of the performance of machine learning algorithms for face mask detection," *2020-5th International Conference on Information Technology* (InCIT), IEEE, 2020.

[20] Naufal, M. Farid, et al, "Comparative Analysis of Image Classification Algorithms for Face Mask Detection," *Journal of Information Systems Engineering and Business Intelligence* 7.1, 2021.

Steven Reynandi Owen, kelahiran kota Bandung tahun 2000. Lulusan SMAK Kalam Kudus Bandung. Menjalani pendidikan di Institut Teknologi Harapan Bangsa sebagai mahasiswa jurusan Media and InternetTechnology (MIT). Tertarik pada bidang IoT, pengembangan perangkat lunak, dan pengembangan Artificial Intelligence.

Yoyok Yusman Gamaliel, kelahiran Ciamis tahun 1974 dan memperoleh gelar Sarjana Teknik dari Universitas Kristen Satya Wacana, dan Master of Engineering dari University of South Australia. Minat penelitian pada analisis data serta pemodelan dan simulasi sistem. Saat ini aktif sebagai staf pengajar di Program Studi Teknik Komputer Institut Teknologi Harapan Bangsa.

Herry Imanta Sitepu , menempuh pendidikan S1 di Teknik Elektro ITB dan lulus tahun 1999, dan memperoleh gelar magister dan doktor di jurusan yang sama di ITB. Sejak tahun 2006 aktif sebagai pengajar di Prodi Sistem Komputer ITHB. Minat penelitian : computer networking, programming dan distributed system.