PENERAPAN ALGORITMA *MOBILENET SINGLE SHOT DETECTOR* UNTUK DETEKSI API DAN ASAP BERPOTENSI KEBAKARAN PADA CITRA HUTAN

SKRIPSI

Diajukan Sebagai Salah Satu Syarat Untuk Mendapatkan Gelar Sarjana Komputer (S.Kom) Program Studi Informatika

ABD SALAM

105841110220

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MAKASSAR

2024

UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK
GEDUNG MENARA IQRA LT. 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بسم الله الرحمن الرحيم

PENGESAHAN

Skripsi atas nama Abd. Salam dengan nomor induk Mahasiswa 105841110220, dinyatakan diterima dan disahkan oleh Panitia Ujian Tugas Akhir/Skripsi sesuai dengan Surat Keputusan Dekan Fakultas Teknik Universitas Muhammadiyah Makassar Nomor : 0008/SK-Y/55202/091004/2024, sebagai salah satu syarat guna memperoleh gelar Sarjana Informatika pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar pada hari Sabtu tanggal 26 Agustus 2024.

Makassar, 21 Safar 1446 H / 26 Agustus 2024 M

Panitia Ujian :

1. Pengawas Umum
   a. Rektor Universitas Muhammadiyah Makassar
      Dr. Ir. H. Abd. Rakhim Nanda, ST., MT., IPU. .....
   b. Dekan Fakultas Teknik Universitas Hasanuddin
      Prof. Dr. Eng. Muhammad Isran Ramli, ST., MT. .....
2. Penguji
   a. Ketua : Dr.Ir.Hj.Hafsah Nirwana, ST., MT. .....
   b. Sekertaris : Fahrim Irhamna Rahman, S.Kom., MT. .....
3. Anggota : 1. Muhyiddin A.M. Hayat, S.Kom., MT. : .....
   2. Titin Wahyuni, S.Pd, MT. : .....
   3. Lukman Anas, S.Kom., MT. : .....

Mengetahui :

Pembimbing I	Pembimbing II
Rizki Yusliana Bakti, S.T.,M.T	**Lukman, S.Kom.,M.T**

Dekan

Dr. Ir. Hj. Nurnawaty, ST., MT., IPM.
NBM. 795 108

ABSTRAK

ABD SALAM PENERAPAN ALGORITMA MOBILENET SINGLE SHOT DETECTOR UNTUK DETEKSI API DAN ASAP BERPOTENSI KEBAKARAN PADA CITRA HUTAN (dibimbing oleh Rizki Yuliana Bakti ST.,MT dan Lukman S.Kom.,MT). Penelitian ini bertujuan untuk merancang sistem deteksi api dan asap berpotensi kebakaran pada citra hutan menggunakan algoritma MobileNet Single Shot Detector (SSD). Dalam penelitian ini, model MobileNet SSD dikembangkan dan dievaluasi untuk mendeteksi objek api dan asap pada citra hutan. Hasil pengujian menunjukkan bahwa model yang dibuat memiliki tingkat kesalahan rendah, dengan nilai Localization Loss sebesar 1,987% dan Classification Loss sebesar 4,903%. Model ini juga mencapai nilai Mean Average Precision (mAP) sebesar 31,5% dan Average Recall sebesar 56,6%, yang mengindikasikan kemampuan deteksi yang cukup baik. Dengan demikian, algoritma MobileNet SSD dapat dianggap efektif dalam mendeteksi api dan asap pada citra hutan, meskipun masih terdapat ruang untuk pengembangan lebih lanjut, terutama dalam meningkatkan ukuran dataset dan efisiensi komputasi.

Kata Kunci: Deteksi api dan asap, MobileNet SSD, Citra hutan

Abstract

ABD SALAM PENERAPAN ALGORITMA MOBILENET SINGLE SHOT DETECTOR UNTUK DETEKSI API DAN ASAP BERPOTENSI KEBAKARAN PADA CITRA HUTAN (dibimbing oleh Rizki Yuliana Bakti ST.,MT dan Lukman S.Kom.,MT).*his study aims to design a fire and smoke detection system for forest images using the MobileNet Single Shot Detector (SSD) algorithm. In this research, the MobileNet SSD model was developed and evaluated for detecting fire and smoke objects in forest images. The test results indicate that the model has a low error rate, with a Localization Loss of 1.987% and a Classification Loss of 4.903%. The model also achieved a Mean Average Precision (mAP) of 31.5% and an Average Recall of 56.6%, indicating reasonably good detection capability. Therefore, the MobileNet SSD algorithm can be considered effective in detecting fire and smoke in forest images, although there is room for further improvement, particularly in expanding the dataset size and computational efficiency.*

Keywords*: Fire and smoke detection, MobileNet SSD, Forest images*

KATA PENGANTAR

Segala Puji bagi Allah SWT atas limpahan rahmat, kesehatan dan kekuatannya sehingga skripsi dengan judul **"PENERAPAN ALGORITMA *MOBILENET SINGLE SHOT DETECTOR* UNTUK DETEKSI API DAN ASAP BERPOTENSI KEBAKARAN PADA CITRA HUTAN"** ini dapat penulis selesaikan sebagaimana salah satu syarat untuk penyusunan Skripsi Program Studi Informatika. Shalawat dan junjungan Nabi Beşar Muhammad SAW sebagai uswatun hasanah dan rahmatan lil alamin.

Dalam penyusunan berbagai hambatan dan keterbatasan dihadapi oleh penulis mulai dari tahap persiapan sampai dengan penyelesaian tulisan, namun berkat bantuan bimbingan dan kerja sama berbagai pihak, hambatan dan kesulitan tersebut dapat teratasi.

Dengan segala kerendahan hati, saya selaku penulis ingin mengucapkan terima kasih kepada semua pihak telah membantu dan memberikan bimbingan baik secara langsung maupun tidak langsung. Dan pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT, yang telah memberikan petunjuk, kekuatan, dan rahmat-Nya selama proses penulisan.
2. Kedua orang tua tercinta yang telah memberikan kasih sayang, doa, serta dukungan yang tiada henti. Terima kasih atas segala pengorbanan, motivasi, dan cinta yang selalu kalian berikan sepanjang perjalanan hidup saya. Tanpa doa dan dukungan kalian, saya tidak akan bisa mencapai titik ini. Kehadiran dan cinta kalian adalah sumber inspirasi dan kekuatan bagi saya.
3. Bapak Prof. Dr. H. Ambo Asse, M.Ag., sebagai Rektor Perguruan Tinggi Universitas Muhammadiyah Makassar
4. Ibu Dr. Ir. Hj. Nurnawaty, ST., MT., Selaku Dekan Fakultas Teknik Universitas Muhammadiyah Makassar
5. Bapak Muhyiddin A M Hayat, S.Kom, M.T., Selaku Ketua Prodi Informatika, Fakultas Teknik Universitas Muhammadiyah Makassar.

6. Ibu Rizki Yusliana Bakti, S.T., M.T., Selaku Dosen Pembimbing I yang telah banyak memberikan bimbingan dan bantuannya.
7. Bapak Lukman, S. Kom., M.T., Selaku Dosen Pembimbing II yang telah banyak memberikan bimbingan dan bantuannya.
8. Seluruh Dosen dan Staf Fakultas Teknik Universitas Muhammadiyah Makassar.
9. Teman-teman kelas C Program Studi Informatika Angkatan 2020.

Mengingat keterbatasan dan kemampuan penulis tentu skripsi ini masih terdapat banyak kesalahan. Untuk itu penulis mengharapkan kritik dan saran yang membangun, dan semoga skripsi ini dapat bermanfaat bagi yang membacanya. Akhirnya teriring doa dan harapan semoga segala bantuan yang telah diberikan baik materi maupun moral mendapat imbalan disisi Tuhan Yang Maha Esa dan bermanfaat bagi kita semua, Aamiin.

Makassar, 1 Mei 2024

ABD SALAM

DAFTAR ISI

ABSTRAK ii
Abstract iii
KATA PENGANTAR iv
DAFTAR ISI vi
DAFTAR TABEL viii
DAFTAR GAMBAR ix
DAFTAR ISTILAH xi
BAB 1 PENDAHULUAN 1
A. Latar Belakang 1
B. Rumusan Masalah 2
C. Tujuan Penelitian 3
D. Manfaat Penelitian 3
E. Ruang Lingkup 3
F. Sistematika Penulisan 4
BAB II TINJAUAN PUSTAKA 6
A. Landasan Teori 6
B. Penelitian Terkait 16
C. Kerangka Pikir 19
BAB III METODOLOGI PENELITIAN 20
A. Tempat dan Waktu Penelitian 20
B. Alat dan Bahan 20
C. Perancangan Sistem 20
D. Teknik Analisis Data 23
E. Teknik Analisis Data 25
BAB IV HASIL DAN PEMBAHASAN 26
A. Pengumpulan data 26
B. Image Annotation 26
C. Preprocessing 28

D. Pembuatan Model *MobileNet* 31
E. Training Model 34
F. Evaluasi Model 37
G. Perancangan Sistem 43
H. Hasil Output Rancangan 45
BAB V KESIMPULAN DAN SARAN 46
A. Kesimpulan 46
B. Saran 46
DAFTAR PUSTAKA 47
LAMPIRAN 51

DAFTAR TABEL

Tabel 1 Simbol Flowchart 12
Tabel 2 Confusion Matrix.....14
Tabel 3 Hyperparameter 33

DAFTAR GAMBAR

Gambar 1 Kebakaran Hutan Gunung Bawakaraeng 6
Gambar 2 Arsitektur Mobilenet V2.....8
Gambar 3 Arsitektur Mobilenet V2.....9
Gambar 4 Arsitektur Mobilenet V2.....10
Gambar 5 Arsitektur Mobilenet V2.....11
Gambar 6 Arsitektur Mobilenet V2.....19
Gambar 7 Flowchart pembuatan model Mobilnet SSD.....21
Gambar 8 Flowchart Rancangan Sistem 22
Gambar 9 Flowchart Rancangan Sistem 26
Gambar 10 Flowchart Rancangan Sistem 31
Gambar 11 Mean Average Precision (Map) dan Average Recall 36
Gambar 12 Localization loss 42
Gambar 13 Classification loss 42
Gambar 14 Tampilan Rancangan Interface 43
Gambar 15 Upload Image to Detect.....44
Gambar 16 Upload Image to Detect.....44
Gambar 17 Upload Image to Detect.....45
Gambar 18 Hasil Output Deteksi 45

DAFTAR LAMPIRAN

Lampiran 1 Surat Permohonan Penelitian 51
Lampiran 2 Surat Izin Penelitian LP3M.....52
Lampiran 3 Data Mentah.....54
Lampiran 4 Data Hasil Cluster 55
Lampiran 5 Source Code.....56
Lampiran 6 Hasil Validasi 75

DAFTAR ISTILAH

MobileNet SSD (Single Shot Detector)	Model deteksi objek yang ringan dan cepat menggunakan arsitektur *MobilNet* untuk ekstraksi fitur dan metode SSD untuk deteksi objek dalam suatu proses komputer
API (Application ProgrammingInterface)	Kumpulan protokol dan alat yang digunakan untuk membangun aplikasi perangkat lunak. API memungkin komunikasi antara dua komponen perangkta lunak yang berbeda.
Bounding Box	Kotak pembatas yang digunakan dalam deteksi objek untuk menandai lokasi objek dalam gambar.
Confusion Matrix	Matrix yang digunakan untuk mengukur performa model klasifikasi dengan menampilkan jumlah prediksi yang benar dan yang salah dilakukan oleh model dalam bentuk tabel.
Dataset	Kumpulan data yang digunakan untuk melatih dan menguji model dalam koteks ini, dataset terdiri dari gambar-gambar yang dianotasi dengan label api dan asap
Feature Learning Process	Proses dimana model belajar mengenali model atau fitur penting dalam data yang diperlukan untuk membuat prediksi yang akurat
Flask	*Framework web Python* yang ringan digunakan untuk membuat aplikasi web termasuk API
Image Annotation	Proses penandaan atau pemberian label pada gambar untuk menandai objek atau fitur

	tertentu. Anotasi ini diperlukan agar model dapat belajar mengenali objek yang di tandai.
Image Augmentation	Teknik manipulasi citra untuk menghasilkan variasi baru dari dataset, seperti rotasi, zoom dan flipping, dengan tujuan meningkatkan kemampuan generalisasi model
Training	Proses melatih model menggunakan dataset yang telah disiapkan untuk belajar mengenali pola-pola tertentu dalam data. Model akan di update terus menerus hingga mencapai performa yang optimal.
Preprocessing	Tahap pengelohana awal pada data yang bertujuan untuk mempersiapka data sebelum digunakan dalam proses training model. Contohnya termasuk resizing gambar.

BAB 1
PENDAHULUAN

A. Latar Belakang

Kebakaran hutan dan lahan merupakan isu yang serius, terutama di negara-negara dengan iklim tropis seperti Indonesia. Kebakaran ini disebabkan oleh tindakan manusia yang sengaja maupun kelalaian. Tindakan sengaja meliputi pembakaran hutan untuk membuka lahan baru, sedangkan kelian termasuk membakar sampah tanpa pengawasan dan membuang puntung rokok sembarangan. Setelah terjadi kebakaran, seringkali api sulit dikendalikan dan menjalar sesuai arah angin. Dampak dari kebakaran hutan sangat merugikan manusia hewan, dan tumbuhan, seperti pemanasan global, perubahan iklim, tanah longsor, dan banjir.(Saputra & Faisal Adhinata, 2023)

Kebakaran hutan hutan di Indonesia juga memiliki dampak luas bagi lingkungan. Emisi karbon dioksida (CO2e) akibat kebakaran mencapai lebih dari 5,9 juta ton dan luas area hutan yang berkurang secara signifikan. Negara Indonesia mengalami kerugian materiil sebesar 200 trilyun rupiah akibat kebakaran tersebut. Selain itu,bencana kabut asap yang ditimbulkan oleh kebakaran bisa berdampak buruk pada kesehatanasyarakat.(Nabilah Muhamad, 2023)

Pemerintah Indonesia perlu meningkatkan pengawasan dan pengendalian kebakaranutan dengan cara meningkatkan kesadaran masyarakat tentang pentingnya menjaga kelestarian hutan serta meningkatkan upaya dalam pengawasan dan pengendalian kebakaran. Pengelolaanhan yang efektif dapat membantu mengurangi risiko terjadinya kebakaran hutan. Salah satu langkah yang dapat diambil adalah mengurangi penggunaan lahan untuk pertanian dankebunan, sambil meningkatkan alokasi lahan untuk tujuan konservasi lingkungan.

Water Bombing, yaitu metode memadamkan kebakaran hutan dengan menyiramkan air ke area yang terbakar, merupakan cara efektif untuk meredam api besar dan berperan dalam pencegahan kebakaran. Pemadaman manual juga seringunakan dengan cara menyiram air langsung pada titik api menggunakan

selang, namun perlu diwaspadai karenaensi bahaya terhadap petugas dari asap dan api.(Wening, 2019)

Citra hutan Indonesia adalah gambar atau citra yang digunakan untuk mendeteksi dan memau kebakaran hutan di Indonesia. Citra ini bisa doleh melalui berbagai metode seperti satelit, drone, atau kamera dipasang di lokasi hutan. Citra ini berguna dalam mendeteksi tit api dan asap dari kebakaran hutan serta memantau luas area terbakar serta proses pemadaman. Penggunaan citra sangat penting dalam upaya mencegah dampak bur dari kebakaran tersebut. Dengan menggunakan citra tersebut petugas pemadam bisa mendeteksi titik api dengan cepat sehingga tindakan pencegahan bisa dilakukan tepat waktu Selain itu juga membantu analisis dampak serta strategi pencegahan di masa depan.(Abror, 2019)

Dalam beberapa penelitian sebelumnya, algoritma MobileNet SSD telah digunakan untuk mendeteksi objek yang bergerak, seperti manusia pada video dan jenis kendaraan. Algoritma ini telah menunjukkan hasil yang baik dengan tingkat akurasi deteksi yang tinggi (misalnya 90% untuk detsi manusia pada video) dan tingkat akurasi deteksi jenis kendaraan yang tinggi (misalnya 52,48% deteksi 10 genera ikan laut).(Muharram et al., 2021)

Dalam penelitian ini, kami akan menganalisis penerapan algoritma MobileNet SSD dalam mendeteksi api dan asap potensial kebakaran pada citra hutan. Kami akan menggunakan dataset citra hutan yang telah dikumpulkan sebelumnya serta melakukan evaluasi akurasi algoritma dengan metode yang sesuai. Harapan dari penelitian ini adalah dapat membantu pengembangan sistem deteksi api dan asap potensial kebakaran secara efektif dan efis serta berkontribusi dalam upaya pencegahan dan pengurangan dampak kebakaran hutan.

B. Rumusan Masalah

Berdasarkan pada latar belakang masalah yang telah diuraikan, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana rancangan sistem untuk deteksi api dan asap?
2. Bagaimana performa algoritma *MobileNet Single Shot Detector* untuk deteksi api dan asap pada citra hutan?

C. Tujuan Penelitian

Tujuan dari penulisan ini yaitu :

1. Membuat rancangan Sistem deteksi api dan asap menggunakan algoritma *MobileNet Single Shot Detector*.
2. Mengevaluasi performa algoritma *MobileNet Single Shot Detector* untuk deteksi api dan asap pada citra hutan

D. Manfaat Penelitian

1. Bagi Masyarakat
   a. Penelitian ini diharapkan dapat meningkatkan kesadaran masyarakat dan respons pihak berwenang terhadap bencana alam, memungkinkan evakuasi dini dan penanganan darurat yang lebih efektif serta berdampak positif terhadap perlindungan lingkungan dan kehidupan manusia secara keseluruhan.
2. Bagi Mahasiswa
   a. Penelitian ini diarahkan untuk memberikan kontribusi yang signifikan dalam bidang pencegahan kebakaran hutan melalui kemampuan deteksi api dan asap algoritma *MobileNet Single Shot Detector* pada citra hutan. Dengan menyelidiki bagaimana *MobileNet Single Shot Detector* berkinerja dalam mendeteksi api dan asap.
   b. Penelitian ini memberikan pemahaman mendalam yang dapat digunakan untuk meningkatkan efektivitas upaya deteksi bencana alam. Selain itu, evaluasi performa algoritma *MobileNet Single Shot Detector* memberikan pandangan yang holistik terhadap kemampuannya dalam konteks yang sama.

E. Ruang Lingkup

1. Penelitian ini akan difokuskan pada Lereng Gunung Bawakaraeng yang beralamat di wilayah Parangmaha, Lingkungan Topidi, Kelurahan Bontolureng Kecamatan Tinggimoncong, Kabupaten Gowa Sulawesi Selatan.
2. Penelitian ini akan memberikan kontribusi pada upaya pencegahan dan penanggulangan dampak kebakaran hutan di Indonesia.

3. Penelitian akan menggunakan algoritma *MobileNet Single Shot Detector* untuk deteksi api dan asap pada citra hutan.

F. Sistematika Penulisan

Pada bagian ini diuraikan sistematika penulisan dari penyusunan Proposal ini yang disajikan secara sistematis sebagai berikut:

BAB I : PENDAHULUAN

Bab ini memberikan gambaran mengenai latar belakang masalah, rumusan permasalahan, batasan masalah, tujuan penelitian dan manfaat penelitian serta sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Berisikan kerangka pikir, landasan teori yang teridiri dari *Mobilenet SSD*, *Convolutional Neural Network*, Flowchart, Phyton, OpenCV, *Confusion Matrix,* Flask, Tensorflow.

BAB III : METODE PENELITIAN

Bab ini berisikan Waktu dan Tempat Penelitian, Bahan dan Alat Penelitian, Jenis dan Variabel Penelitian, Pengumpulan Data dan Analisis Data.

BAB IV : HASIL PENELITIAN DAN PEMBAHASAN

Bab ini membahas tentang perancangan solusi, Analisa dan validasi data serta hasil dan pembahasan.

BAB V : KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan dan saran dari penelitian yang telah dilakukan.

BAB II
TINJAUAN PUSTAKA

A. Landasan Teori

1. Kebakaran Hutan

Kebakaran hutan dapat diartikan sebagai kejadian dimana api melahap bahan bakar bervegetasi, yang terjadi di kawasan hutan dan non-hutan yang menjalar secara bebas dan tidak terkendali. Kebakaran hutan ialah terbakarnya sesuatu yang menimbulkan bahaya atau mendatangkan bencana. Kebakaran dapat terjadi karena pembakaran yang tidak dikendalikan, karena proses spontan alami, atau karena kesengajaan.(Ari Kukuh Sentanu et al., 2021)

Gambar 1 Kebakaran Hutan Gunung Bawakaraeng

Sumber: (Pranata, 2019)

Dampak yang ditimbulkan oleh kebakaran hutan mencakup beberapa hal, seperti peningkatan emisi gas karbon dioksida ke atmosfer berdampak pada satwa liar dan kerusakan habitat tanaman baik karena terbakar atau terjebak asap,

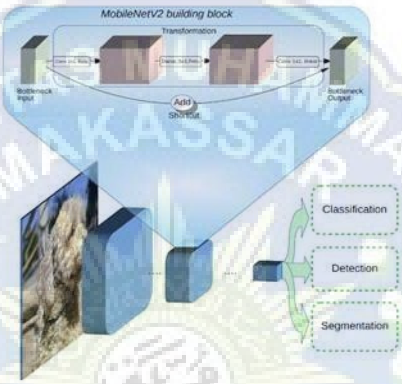
meningkatnya jumlah penderita penyakit infek saluran pernafasan (ISPA), gangguan aktivitas masyarakat di bidang ekonomi, pendidikan, dan lain-lain. Kebakaranutan sering kali baru terdeteksi ketika api sudah mulai membesar dan meluas, sehingga upaya penanggulangan sudah terlambat sulit untuk memadamkan api yang telah meluas.

2. MobileNet SSD

SSD-*MobileNet* merupakan kombinasi model *MobileNet* yang digunakan untuk ekstraksi fitur dan model SSD digunakan untuk deteksi objek dalam citra. *MobileNet* adalah model arsitektur streamlined yang menggunakan *depthwise separable convolutions* dalam pembentukan jaringan saraf tiruan mendalam. *Depthwise separable convolution* terdiri dari dua lapisan yaitu *depthwise convolution* yang diterapkan pada citra input untuk menerapkan filter, dan pointwise convolution digunakan untuk menggabungkan hasil *output* dari *depthwise convolution* agar menghasilkan komputasi yang lebih ringan. SSD adalah jaringan konvolusi berbasis *feed forward* yang menghasilkan kotak prediksi dan skor prediksi objek dalam citra. SSD menggunakan lapisan fitur konvolusi untuk melakukan prediksi deteksi objek dalam citra dengan ukuran lapisan fitur tersebut secara bertahap berkurang sehingga dapat mendeteksi objek dengan ukuran bervariasi.(Syahputra, 2023)

Kemacetan dalam *MobileNetV2* terjadi saat lapisan-lapisan di dalamnya mengodekan input dan output perantara untuk merangkum kemampuan model dalam mentransformasikan konsep dari tingkat yang lebih rendah, misalnya piksel, menjadi deskriptor yang lebih tinggi seperti kategori gambar.

Dengan menggunakan koneksi residual tradisional, pintasan memungkinkan pelatihan yang lebih cepat dan akurasi yang lebih baik. *MobileNetV2* tetap memanfaat *depthwise dan pointwise convolution.* Selain itu, dua fitur baru ditambahkan ke MobileNetV2, yaitu: 1) *linear bottleneck*, dan 2) *shortcut connections* antar *bottlenecks* struktur dasar dari arsitektur ini ditunjukkan.



Gambar 2 *Arsitektur Mobilenet V2*

Sumber: (Ekoputris, 2018)

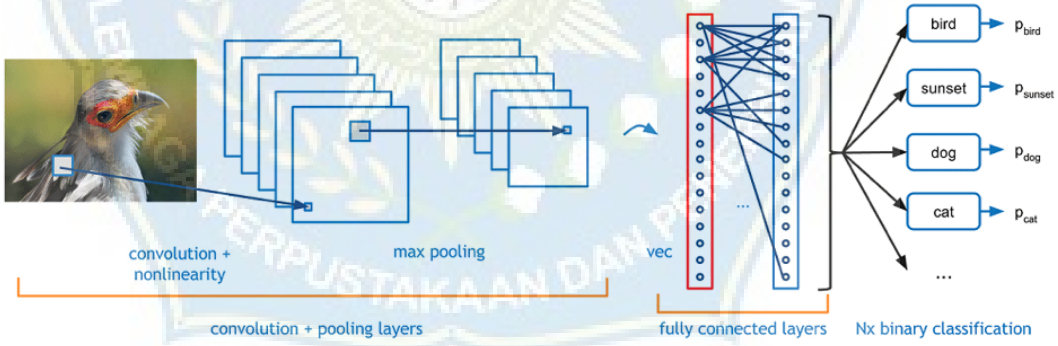
Pada bagian *bottleneck* terdapat *input* dan *output* antara model sedangkan lapisan atau layer bagian dalam meng-enkapsulasi kemampuan model untuk mengubah input dari konsep tingkat yang lebih rendah (piksel) ke deskriptor tingkat yang lebih tinggi.

Pada bagian *bottleneck* terdapat koneksi input dan output antara model, sementara lapisan-lapisan dalamnya memuat kemampuan model untuk mentformasi input dari konsep tingkat yang lebih rendah (piksel) menjadi deskriptor tingkat yang lebih tinggi. Seperti halnya koneksi residu pada CNN tradal, *shortcut* antar *bottlenecks* memungkinkan pelatihan yang lebih cepat dan akurasi yang lebih baik. Arsitektur *MobileNetV2* terdiri dari blok bangunan dasar yang menggunakan konvolusi dengan kedalaman pemisahan dan residu. Terdapat 32 filter pada lan konvolusi penuh awal diikuti oleh 19 lapis bottleneck residual.

Arsitektur ini dapat disesuaikan dengan resolusi gambar input dan pengganda lebar sebagai hyperparameter yang dapat disetel sesuai kebutuhan akurasi atau kinerja. Biaya komputasi jaringan berkisar antara 7 hingga 585 juta perkalian-tambahan dengan ukuran model bervariasi antara 1,7 hingga 6,9 juta parameter. *MobileNetV2* menggunakan teknik seperti *depthwise separable convolution, linear bottleneck, dan shortcut connection* untuk menghasilkan model yang efisien namun tetap memiliki kinerja baik.(Syahputra, 2023)

3. CNN

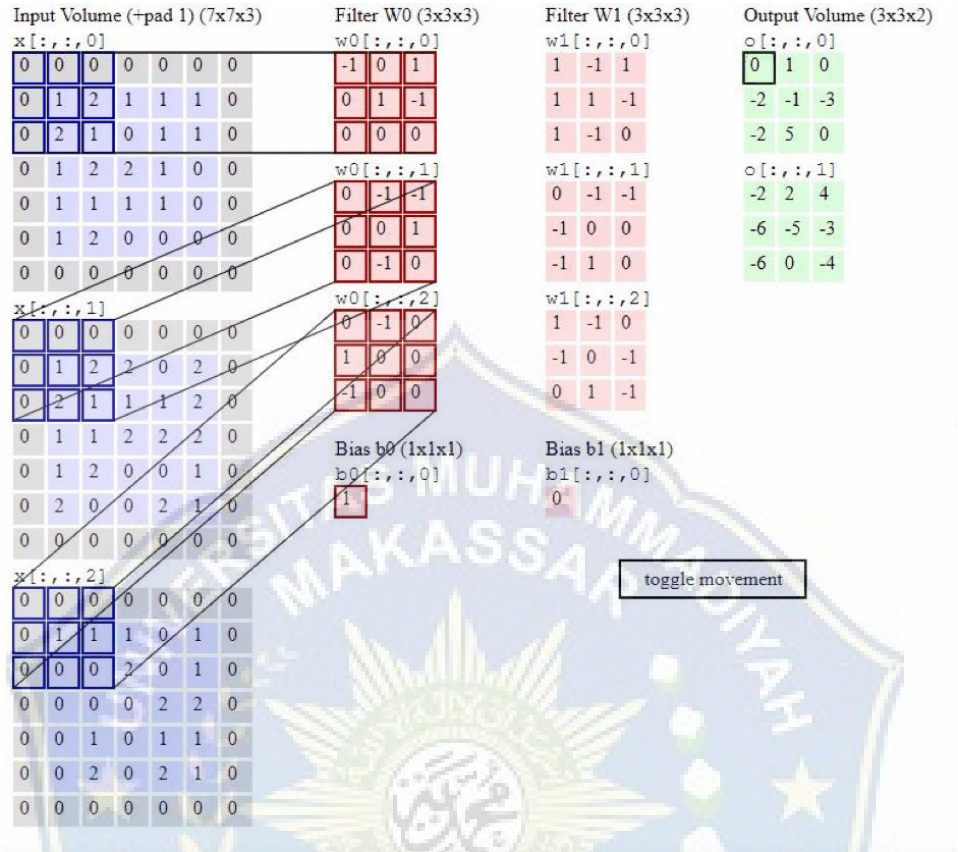
Convolution Neural Network merupakan jenis *Neural Network* atau jaringan saraf tiruan yang digunakan dalam pengolahan data pada sistem cerdas seperti pengolah citra digital. Metode CNN banyak digunakan dalam pengenalan citra tertentu. Proses pengenalan citra pada CNN melibatkan *layer-layer* dimana setiap layernya terhubung satu sama lainnya. Pada *Fully Connected layer* setiap unit terhubung dengan semua unit dari *layer* sebelumnya, sedangkan *Convolution layer* menghubungkan masing-masing unit dengan bobot tertentu di daerah tertentu dari *layer* sebelumnyasebelumnya. (Mamuriyah & Sumantri, 2022)



Gambar 3 *Arsitektur Mobilenet V2*

(Nurmalasari, 2019)

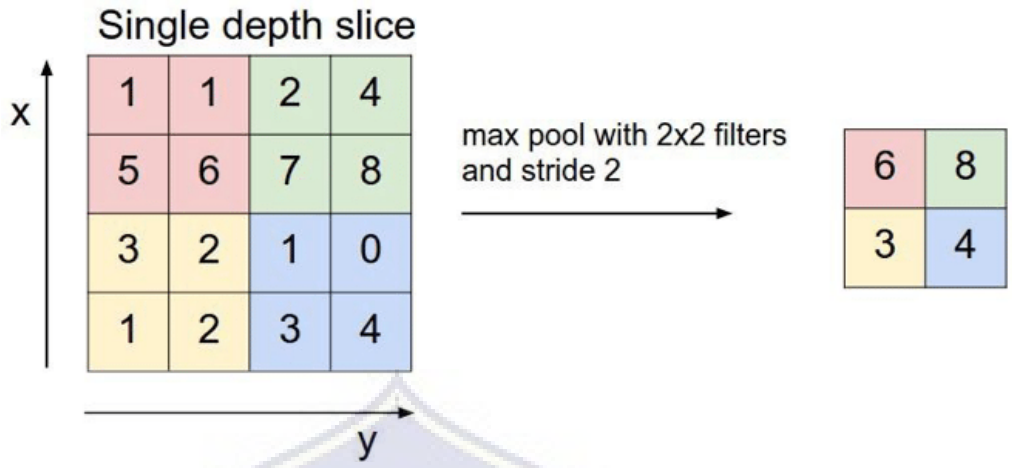
Layer pertama dari CNN adalah *convolutional layer* dengan ukuran 7 x 7 x 3. Filter-filter ini akaneser ke seluruh bagian gambar, dan pada setiap pergeseran, operasi "dot" dilakukan antara input dan nilai filter untuk menghasilkan output *(activation map)*. (Adarrani et al., 2022)



Gambar 4 *Arsitektur Mobilenet V2*

Sumber: (Trivusi, 2022)

Setelah *convolutional layer* terdapat *pooling layer*. Terdiri dari sebuah filter dengan ukuran dan stride tertentu yang akan bergeser pada seluruh activation map. Pooling yang digunakan adalah *max* dan *average pooling*. Misal kita gunakan max pooling 2 x 2 dengan stride 2, maka disetiap pergeseran filter, nilai maksimum pada area 2 x 2 pixel tersebut yang akan dipilih, sedangkan average pooling akan memilih nilai rata-ratanya.



Gambar 5 *Arsitektur Mobilenet V2*

Sumber(Trivusi, 2022)

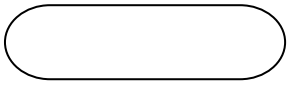
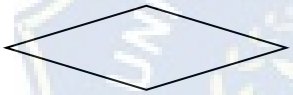
Pooling layer digunakan untuk mempercepat komputasi karena parameter yang harus di update semakin sedikit dan mengatasi *overfitting.*(Adarrani et al., 2022)

4. Flowchart

Flowchart atau diagram alir adalah jenis diagram yang digunakan untuk merepresentasikan algoritma atau langkah-langkah instruksi secara berurutan dalam sebuah sistem. *Flowchart* membantu analis sistem menjelaskan logika sistem kepada *programmer* sebagai bukti dokumentasi. Simbol-simbol digunakan dalam *flowchart* untuk mewakili setiap proses danis penghubung digunakan untuk menghubungkan proses satu ke proses berikutnya (Rosaly & Prasetyo, 2020)

Flowchart memberikan gambaran jelas tentang urutan proses dalam sebuah sistem dan memudahkan penambahan proses baru di masa depan. Setelah pembuatan *flowchart* selesai, programmer dapat menerjemahkan desain logis tersebut menjadi program dengan bahasa pemrograman yang telah disepakati.

Tabel 1 Simbol *Flowchart*

Simbol	Nama	Fungsi
	*Terminator*	Permulaan/akhir program
	Garis Alir (*Flow Line*)	Arah aliran program
	Proses	Proses perhitungan / proses pengolahan data
	*Input/Output Data*	Proses *input / output* data, parameter informasi
	*Decision*	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya

5. Python

Python adalah sebuah bahasa pemrograman komputer yang sering digunakan untuk membuat situs web, aplikasi, serta melakukan analisis data. Bahasa ini termasuk dalam kategori bahasa pemrograman umum yang bisa digunakan untuk berbagai keperluan. Kelebihan *Python* antara lain adalah sintaksisnya yang mudah dipahami, adanya komunitas besar yang aktif, serta dukungan pustaka dan kerangka kerja yang luas. Karena kemudahan belajar dan fleksibilitasnya, *Python* menjadi pilihan populer baik untuk pemula maupun pengembang berpengalaman. Banyak non-programmer seperti akuntan dan ilmuwan juga menggunakan *Python* dalam tugas-tugas sehari-hari. (Alfarizi et al., 2023)

Banyak programmer dan peneliti memilih menggunakan *Python* karena fleksibilitnya dalam berbagai bidang pengembangan seperti aplikasi *web, desktop, IoT,* serta kemampuannya mengintegrasikan dengan sistem database dan file. Selain itu, keunggulan *Python* juga terletak pada kemampuannya menangani data besar dan perhitungan matematika kompleks sehingga banyak digunakan oleh paneliti.(Rahman et al., 2023)

Ada beberapa alasan mengapa Python menjadi pilihan utama, yaitu:(Rahman et al., 2023)

1. Python dapat berjalan di berbagai platform seperti Windows,

Linux, macOS, Android, Raspberry Pi, dan lain-lain.

2. Python memiliki sintaks yang sederhana dan mirip dengan bahasa Inggris.

3. Sintaks Python memungkinkan penulisan kode yang lebih ringkas dibandingkan dengan bahasa pemrograman lain.

4. Python menggunakan interpreter, sehingga program dapat dieksekusi dengan cepat setelah selesai dibuat.

5. Python mendukung paradigma pemrograman prosedural, berorientasi objek, dan fungsional.

6. OpenCV

OpenCV (Open Source Computer Vision Library) adalah sebuah library open sourceyang dikembangkan oleh intel yang fokus untuk menyederhanakan programingterkait citra digital. Di dalam OpenCV sudah mempunyai banyak fitur, antara lain: pengenalan wajah, pelacakan wajah, deteksi wajah, kalman filtering, dan berbagai jenismetode AI (Artificial Intellegence)dan menyediakan berbagai algoritma sederhana terkait computer visionuntuk low level, OpenCV merupakan open source computer vision libraryuntuk bahasa pemrograman C/C++, dan telah dikembangkan ke Phyton, Java, Matlab.(Rahman et al., 2023)

7. Confusion Matrix

Confusion Matrix adalah matriks yang berisikan mengenai hasil yang diperoleh dari prediksi klasifikasi dan data aktual yang dilakukan oleh sistem klasifikasi. Jadi, confusion matrix berisikan suatu informasi dari sistem yang sudah melakukan perbandingan antara hasil pengelompokan dengan hasil pengelompokan yang diharapkan atau seharusnya keluar. Umumnya kinerja sistem klasifikasi dihitung dengan memerlukan beberapa data yang ada dalam tabel matriks menggunakan 4 (empat) istilah sebagai representasi hasil proses klasifikasi.(Amalina, 2019) Hasil dari proses perhitungan *confusion matrix* yaitu 4 keluaran diantaranya *recall, precision, accuracy,* dan *error rate*. Dapat dilihat pada tabel dibawah ini:

Tabel 2 *Confusion Matrix*

		Prediksi	
Aktual	Negatif	A	C
	Positif	B	D

Keterangan:

a. A = jumlah prediksi yang tepat bersifat negatif.

b. B = jumlah prediksi yang salah bersifat positif.

c. C = jumlah prediksi yang salah bersifat negatif.

d. D = Jumlah prediksi yang tepat bersifat positif.

Beberapa persyaratan yang telah didefinisikan untuk matrik klasifikasi diantaranya sebagai berikut:

a. *Accuracy* merupakan proporsi jumlah prediksi benar. Rumus akurasi adalah:

$$AC = (A + D) / A + B + C + D \quad \text{.....(1)}$$

b. *Recall* atau tingkat positif benar (TP) adalah proporsi kasus positif yang diidentifikasi dengan benar, yang dapat dihitung dengan persamaan:

TP = D / C +D ..... (2)

c. Tingkat positif salah (FP) adalah proporsi kasus negatif yang salah diklasifikasikan sebagai positif, yang dapat dihitung dengan menggunakan persamaan:

FP = B / A+B ..... (3)

d. Tingkat negatif sejati (TN) didefinisikan sebagai proporsi kasus negatif yang diklasifikasikan dengan benar, dapat dihitung dengan menggunakan persamaan:

TN = A / A+B ..... (4)

e. Tingkat negatif palsu (FN) adalah proporsi kasus positif yang salah diklasifikasikan sebagai negatif, yang dihitung dengan menggunakan persamaan:

FN = C / C +D ..... (5)

f. *Precision* (P) adalah proporsi prediksi kasus positif yang benar, yang dihitung dengan menggunakan persamaan:

P = D / B + D ..... (6)

8. Flask

Flask merupakan web framework yang digunakan oleh *Python. Flask framework* tergolong ke dalam *micro-framework* karena tidak membutuhkan *tools* atau *library* tertentu serta memiliki database bawaan. *Flask* menggunakan Jinja *Template Engine dan Werkzeug WSGI ToolKit. Flask* menyusun kategorinya menjadi dua bagian yaitu: *Static File* yang memiliki semua kode status yang dibutuhkan untuk website seperti kode CSS, kode JavaScript dan file gambar, dan Template File yang berisi semua template Jinja termasuk halaman HTML. Dalam mengembangkan aplikasi dari *framework* ini, diperlukan *virtual environment* untuk menampung pustaka yang akan digunakan. (Ningtyas & Setiyawati, 2021)

B. Penelitian Terkait

1. "Pendeteksi Kebakaran Hutan Menggunakan Komunikasi LoRa (Long Range) Wireless Network" (Adelianthi, 2019)

Penelitian ini bertujuan untuk mendeteksi dan memantau tingkat kerawanan terjadinya kebakaran hutan serta memberikan informasi kepada petugas secara real time yang akan ditampilkan pada website. Menggunakan metode penelitian eksperimental dengan pendekatan saintifik dan kuantitatif, pengumpulan data dilakukan melalui observasi dan tinjauan literatur, sementara metode pengujian yang digunakan adalah pengujian black box. Metode perancangan yang diterapkan dalam penelitian ini adalah metode prototipe. Hasil penelitian ini adalah terciptanya sebuah alat yang dapat membantu operator maupun petugas dalam mengetahui kondisi hutan secara real time, terhubung dengan website. Pengujian alat ini menunjukkan bahwa semua fungsi yang diharapkan berhasil beroperasi sesuai dengan yang diharapkan.

2. "Desain Sistem Pendeteksi Kebakaran Hutan Dengan GPS dan Telegram " (Winarno & Awang Joko Mastera, 2023)

Penelitian ini membahas sistem pendeteksi kebakaran hutan yang bekerja secara realtime dan praktis menggunakan teknologi GPS dan Telegram untuk memberikan informasi lokasi kebakaran secara akurat. Metode yang digunakan melibatkan penggunaan sensor api (flame sensor) dan sensor asap (MQ-9) berbasis mikrokontroler untuk mendeteksi api dan asap. Teknologi lain yang digunakan dalam penelitian ini adalah Wireless Sensor Network (WSN) yang terhubung dengan sensor, serta sensor suhu (LM-35) untuk mendeteksi panas di dalam ruangan, yang mengirimkan informasi melalui SMS Gateway menggunakan Arduino. Hasil penelitian sebelumnya menunjukkan bahwa sistem pendeteksi kebakaran dapat menginformasikan kebakaran secara realtime, tetapi tanpa memberikan lokasi tepat dari titik awal api.

3. "Desain Dan Implementasi Dashboard Monitoring Sistem Pendeteksi Kebakaran Hutan Berbasis Lora Dan Web" (Ruldivem et al., 2022)

Penelitian ini menjelaskan tentang sistem deteksi kebakaran hutan yang mampu mentransmisikan data lapangan ke operator melalui koneksi Internet (IoT) secara realtime. Data yang dikirimkan mencakup suhu, kandungan udara/asap di hutan, kondisi api di sekitar perangkat, dan lokasi perangkat itu sendiri, yang semuanya penting untuk memantau keadaan hutan dan melakukan pencegahan dini sebelum kebakaran hutan yang lebih besar terjadi. Metode yang digunakan dalam penelitian ini melibatkan perangkat keras seperti LoRa sebagai receiver, Arduino Uno sebagai mikrokontroler, dan modul GSM (SIM800L) untuk komunikasi. Hasil dari penelitian ini menunjukkan bahwa sistem yang dirancang mampu memberikan informasi penting secara realtime, membantu operator dalam memantau dan mencegah kebakaran hutan dengan lebih efektif.

4. "Sistem Real Time Monitoring Pendeteksi Kebakaran Hutan dan Lahan di Provinsi Riau" (Irawan et al., 2022)

penelitian ini merancang sistem kebakaran dengan penerapan teknologi Internet of Things (IoT) untuk mendeteksi tanda-tanda kebakaran hutan dan lahan dengan lebih cepat. Metode yang digunakan melibatkan pemanfaatan cloud computing untuk penyimpanan dan pendistribusian data, dengan Partikel Argon (Photon) untuk koneksi WiFi yang kuat. Sensor IR Fire Detector digunakan untuk mendeteksi kebakaran melalui pola spektral api, sementara Sensor DHT22 memantau kualitas udara dan kelembaban. Modul GPS Neo 6m digunakan untuk menentukan lokasi kebakaran, dan sensor MQ2 serta Soil Moisture Sensor digunakan untuk mengukur kondisi kadar udara dan kelembaban tanah. Data yang dihasilkan dari detektor kebakaran disimpan dalam database lokal. Hasil penelitian menunjukkan bahwa alat pendeteksi kebakaran ini dapat bekerja dengan baik menggunakan parameter api, suhu, asap, kelembaban udara, dan kelembaban tanah. Partikel Argon mampu menerima input dan membuat koneksi IoT ke web server, memungkinkan pengguna memantau kondisi lahan secara *real time*.

5. "Protype Sistem Pemantauan dan Pendeteksi Kebakaran Hutan dan Lahan Menggunakan Teknologi WSN Berbasis IoT" (Andeskob, 2023)

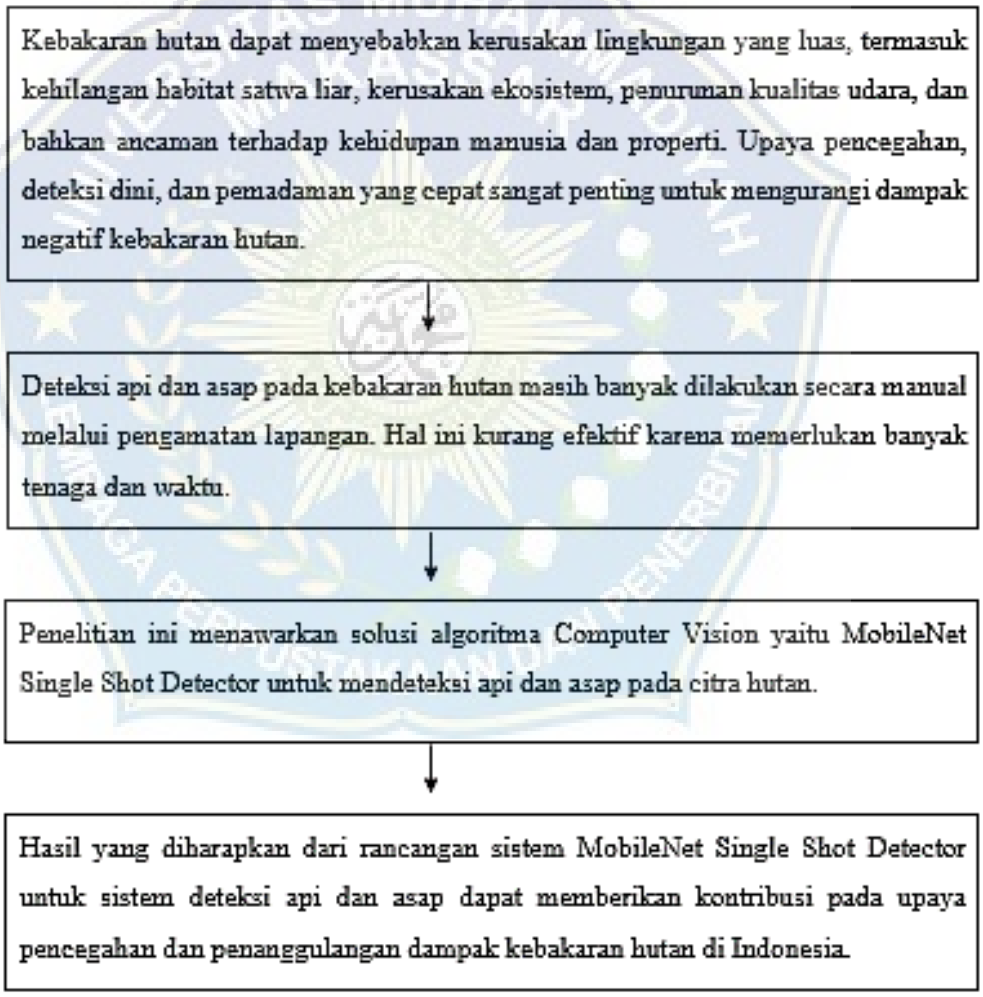
Penelitian ini membahas penggunaan teknologi Wireless Sensor Network (WSN) untuk mendeteksi kebakaran hutan dan memantau suhu, kelembaban, dan konsentrasi asap secara real time. Prototipe yang dikembangkan memanfaatkan platform IoT untuk pengiriman data melalui internet. Sensor yang digunakan meliputi DHT22 untuk suhu dan kelembaban, anemometer untuk kecepatan angin, dan MQ-2 untuk asap. Hasilnya menunjukkan bahwa prototipe mampu memantau kondisi lingkungan hutan dan lahan secara real time, dengan galat suhu 2,41% dan galat kelembaban 17,68% dibandingkan alat BMKG. Meskipun anemometer memiliki delay 4 detik, prototipe tetap efektif dalam mendeteksi perubahan suhu dan kelembaban hingga radius lebih dari 2 meter. Prototipe juga dapat mengirimkan peringatan melalui ponsel Android dan email terkait perubahan kondisi yang terdeteksi.

6. "Peningkatan Metode YOLOv7 Dengan Proses Augmentasi Image Pada Klasifikasi Jenis Kupu-Kupu YOLOv7"(Anggreani, 2023)

Penelitian ini membahas tentang penangkaran kupu-kupu di Taman Nasional Bantimurung Bulusaraung(TN-Babul). Penangkaran kupu-kupu adalah hal yang paling mendominasi pada taman nasional tersebut hingga julukan *The kingdom of butterfly* disebut sebagai TN-Babul. Spesies kupu-kupu cukup banyak yang dimana mencapai 20.000, sekitar 2000 spesies ada di Indonesia dan 557 spesies ada di Pulau Sulawesi. Metode penelitian yang digunakan dalam penelitian ini adalah metode YOLOv7, yang dikombinasikan dengan proses Augmentasi Gambar *(Image Augmentation)* untuk meningkatkan akurasi klasifikasi spesies kupu-kupu. Hasil dari penelitian menunjukkan bahwa dengan menggunakan dataset sebanyak 1000 dan jumlah iterasi 1000, diperoleh nilai mAP *(mean Average Precision)* sebesar 90%. Penelitian ini juga menunjukkan bahwa penggunaan Augmentasi Gambar dapat meningkatkan nilai mAP sebesar 2,97% dibandingkan dengan penelitian sebelumnya.

C. Kerangka Pikir

Dalam penelitian ini, penulis mencoba menawarkan solusi dengan menggunakan Algoritma *MobileNet Single Short Detector* untuk Deteksi Api dan Asap Berpotensi Kebakaran pada Citra Hutan. Diharapkan bahwa rancangan sistem *MobileNet Single Shot Detector* dapat memberikan kontribusi dalam upaya pencegahan dan mitigasi dampak kebakaran hutan di Indonesia.



Gambar 6 *Arsitektur Mobilenet V2*

BAB III
METODOLOGI PENELITIAN

A. Tempat dan Waktu Penelitian

Penelitian dilakukan di Lereng Gunung Bawakaraeng yang beralamat di wilayah Parangmaha, Lingkungan Topidi, Kelurahan Bontolureng Kecamatan Tinggimoncong, Kabupaten Gowa Sulawesi Selatan. Penelitian ini berlangsung selama 3 bulan mulai dari bulan Maret hingga Mei 2024.

B. Alat dan Bahan

1. Alat Penelitian

   Dalam penelitian ini, peneliti menggunakan satu unit Laptop dengan spesifikasi, yaitu *Processor* AMD Ryzen 7 6800H, GPU NVIDIA GeForce RTX 3050, Ram 8 GB dan SSD 512Gb.

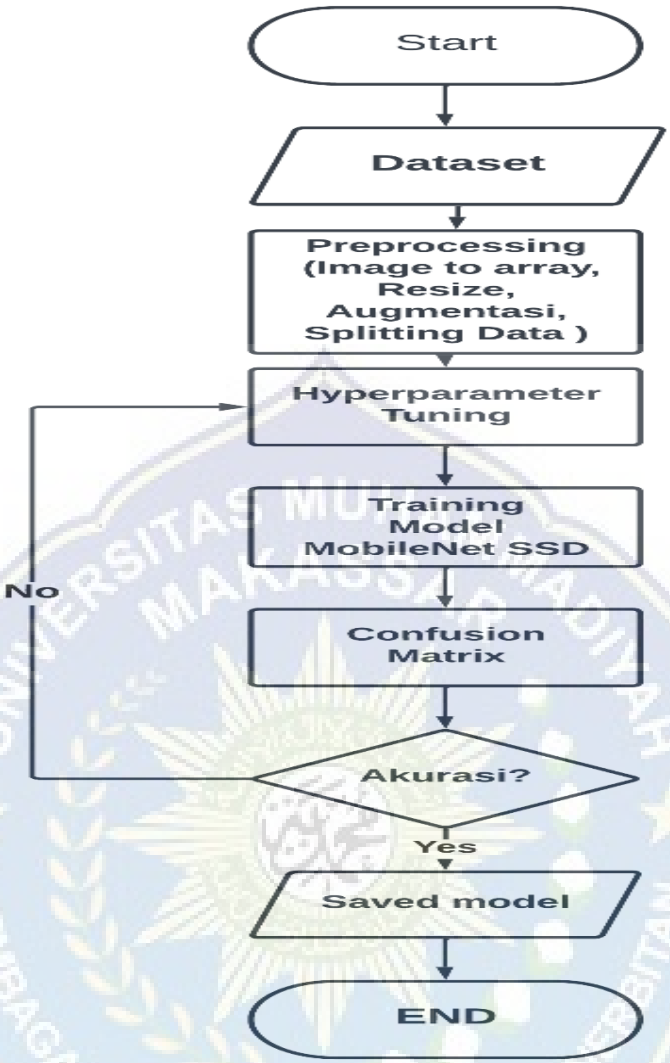
2. Bahan Penelitian

   Dalam penyusunan skripsi ini, peneliti melakukan pengumpulan data *citra* atau foto berupa api dan asap. Dataset berasal dari hasil dokumentasi *handphone* sebanyak 950 foto terdiri dari 475 foto api dan 475 foto asap.

C. Perancangan Sistem

1. Model *MobileNet Single Shot Detector*

   Proses pembuatan model *MobileNet Single Shot Detector* (SSD) melibatkan langkah-langkah yang ditunjukkan dalam diagram alir pada Gambar *Flowchart* pembuatan model *MobileNet SSD*.



Gambar 7 *Flowchart* pembuatan model *Mobilnet SSD*

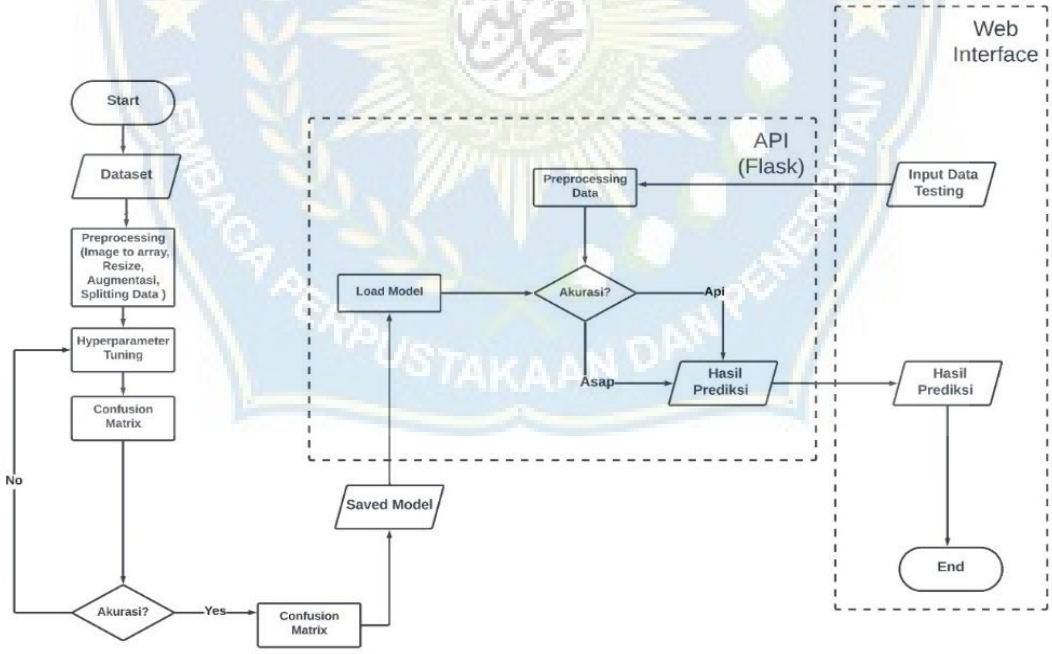
Pada *Flowchart* diatas digambarkan proses pembuatan model *MobileNet SSD* yang tidak jauh berbeda dengan proses pembuatan model *Convolutional Neural Network (CNN)* sebelumnya. Pada dasarnya *Convolutional Neural Network* dan *MobileNet SSD* keduanya merupakan algoritma jaringan syaraf dan memiliki kesamaan arsitektur (susunan algoritma) sehingga proses pembuatan model kedua algoritma tidak jauh berbeda.

Setelah itu, data yang telah melewati *PreProcessing* akan digunakan untuk melatih model dalam proses Training pembuatan model tersebut. Setelah

melalui pelatihan atau training dengan dataset yang telah disiapkan tadi, model akan dinilai performanya menggunakan *Confusion Matrix* untuk menentukan apakah kinerjanya sudah cukup baik atau belum. Jika performa model sudah memadai, maka akan disimpan agar dapat digunakan pada rancangan sistem nantinya. Apabila performa masih kurang optimal, maka tahap Training dilakukan kembali dengan melakukan pengaturan ulang pada *Hyperparameter*.

2. Flowchart Rancangan Sistem

Setelah pembuatan kedua model selesai, nantinya model yang telah di-*Save* akan diterapkan pada rancangan sistem berupa *Interface Web*. Hal ini dilakukan untuk mempermudah proses pengujian dalam mendeteksi adanya api atau asap pada citra hutan yang diuji. Di bawah ini merupakan gambar 3.5 Flowchart Rancangan Sistem yang merupakan Flowchart rancangan sistem *Interface* untuk pengujian model yang telah dibuat.



Gambar 8 *Flowchart* Rancangan Sistem

Pada Flowchart di atas dapat dilihat setelah pembuatan model selesai dan model telah di-*Save*, model akan di-*Load* pada *API* (*Application Programming Interface)* berbasis *Python*, yakni *Flask.* Selain proses *Load* model, di dalam *API* juga dilakukan *PreProcessing* pada data yang diuji (Citra Hutan) yang telah diterima dari *Interface Web.* Setelah dilakukan *PreProcessing*, kemudian citra hutan akan diprediksi oleh model untuk dideteksi apakah adanya api atau asap, setelah itu barulah kemudian hasil prediksi (deteksi api atau asap) dikembalikan ke *Interface Web* sebagai *output*.

D. Teknik Analisis Data

1. Citra

   Pada penelitian ini citra berupa gambar kebakaran hutan yang nantinya akan digunakan ketika proses pembuatan model deteksi api dan asap serta sebagai bahan uji coba pada sistem deteksi yang telah dibuat.

2. Deteksi api dan asap

   Dalam konteks penelitian ini, deteksi api merujuk pada kemampuan sistem untuk mengidentifikasi keberadaan api pada citra hutan. Deteksi api menjadi variabel terikat karena merupakan fokus utama Kemampuan sistem untuk secara akurat dan cepat mendeteksi api pada citra hutan adalah tujuan kritis, dan performa deteksi ini akan diukur dan dievaluasi dalam penelitian. Deteksi asap merujuk pada kemampuan sistem untuk mengidentifikasi partikel-partikel asap yang mungkin menunjukkan adanya kebakaran pada citra hutan.

3. Pengumpulan Data

   Pengumpulan data dilakukan dengan mengambil data dari *Kaggle*. Data ini sebanyak 950 gambar dengan format ekstensi JPG (*Join Photographic Group*) yang terdi dari 2 label yaitu api sebanyak 475 gambar dan asap sebanyak 475 gambar

4. Analisis Data

   Dataset total sebanyak 950 terdiri dari 475 citra api dan 475 citra asap yang telah dikumpulkan nantinya akan melalui beberapa tahap sebelum proses

pembuatan model baik dengan algoritma *MobileNet Single Shot Detector*. Berikut diuraikan dalam beberapa point.

a. Image Annotation

Image annotation adalah proses menambahkan metadata atau label ke dalam gambar digital untuk mengidentifikasi dan menggambarkan objek, fitur, atau informasi tertentu dalam gambar tersebut. Tujuan utama dari image annotation adalah untuk memberikan pemahaman yang lebih baik tentang konten gambar, baik oleh manusia maupun oleh sistem komputer.

Image annotation merupakan tahap penting dalam banyak aplikasi berbasis citra, termasuk pengenalan wajah, deteksi objek, kendaraan otonom, penglihatan komputer medis, dan banyak lagi. Dengan menyediakan data yang telah dianotasi dengan baik, sistem komputer dapat belajar untuk mengenali dan menginterpretasikan gambar dengan lebih baik.

b. Pre-Processing Data

Pada proses ini dilakukan tahap PreProcessing yang bertujuan untuk mempersiapkan data-data sebelum dilakukan proses Training Model. Setiap data harus memiliki dimensi ukuran yang sama, sehingga perlu dilakukan Resize atau penyetaraan dimensi citra. Umumnya gambar di-Resize ke ukuran 320 x 320 pixel. Selain itu juga dilakukan Image Augmentation, proses ini bertujuan untuk membuat variasi baru dari dataset citra atau menambahkan "Feature Complexity", hal ini akan mendukung proses Training Model.

5. Pembagian Data

Dataset yang diperoleh akan dibagi menjadi 2 bagian. Yang pertama, sebagian besar data digunakan untuk training, dan yang kedua, sebagian data digunakan untuk testing. Proporsi data yang digunakan untuk training adalah sebesar 70% atau 665 gambar, dan proporsi untuk testing sebanyak 30% atau 285 gambar dari total jumlah dataset.

6. Analisis Hasil

   Hasil pengujian dianalisis untuk mengevaluasi rancangan sistem yang dibuat, pengolahan data di dalam sistem hinggah keseluruhan rancangan selesai dan dapat mendeteksi api dan asap pada citra hutan yang akan diuji.

7. Penyimpulan

   Berdasarkan hasil pengujian, kesimpulan ditarik mengenai keefektifan sistem deteksi api dan asap berbasis MobileNet SSD dalam bidang pencegahan kebakaran hutan melalui kemampuan deteksi api dan asap pada citra hutan.

E. Teknik Analisis Data

Tahap-tahap yang dilakukan dalam menganalisis sistem ini adalah sebagai berikut:

1. Pengumpulan data: mengumpulkan gambar atau citra yang diterapkan dalam penelitian ini melalui studi literatur.
2. Analisis Sistem: penguraian dari suatu analisis yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan, hambatan, yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan perbaikannya.
3. Menganalisis sistem: merupakan strategi untuk memecahkan masalah dan mengembangkan solusi terbaik bagi permasalahan.
4. Coding adalah menerjemahkan persyaratan logika dari pseudocode atau diagram alur ke dalam suatu bahasa pemrograman baik huruf, angka, dan simbol yang membentuk

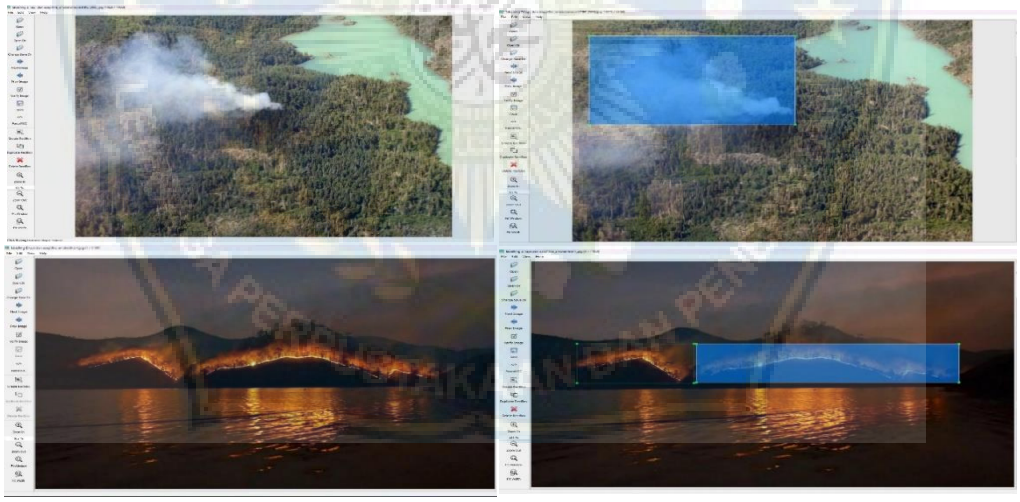
BAB IV
HASIL DAN PEMBAHASAN

A. Pengumpulan data

Pada pengumpulan data, penulis mengumpulkan data dari lokasi penelitian. Data ini sebanyak sebanyak 950 gambar dengan format ekstensi JPG (*Joint Photographic Group*) yang terdi dari 2 label yaitu api sebanyak 475 gambar dan asap sebanyak 475 gambar.

B. Image Annotation

Image annotation adalah proses menandai atau menunjukkan lokasi dan jenis objek dalam suatu gambar. Pada penelitian ini kami menandai setiap objek api dan asap yang ada pada setiap gambar, dimana nantinya model akan belajar mengenali objek yang telah ditandai dalam gambar. Pada gambar 9 diperlihatkan gambar sebelah kiri adalah gambar awal dan gambar kanan merupakan gambar yanbg telah dilakukan Image Annotation.



Gambar 9 *Flowchart* Rancangan Sistem

Gambar yang telah dianotasikan akan menghasilkan file berekstensi XML (Extensible Markup Language) yang berisi rincian informasi dari gambar yang dianotasi. Contoh file XML hasil anotasi dapat dilihat pada gambar xx. Pada file XML tersebut terdapat informasi seperti nama file , jenis kelas (api atau asap) ukuran gambar dalam piksel serta posisi kotak anotasi pada program dibawah ini

```
<annotation>
    <folder>fire</folder>
    <filename>1.jpg</filename>
<path>D:\Skripsi\fire_smoke_data\fire_smoke\fire\1.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>955</width>
        <height>585</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>fire</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>97</xmin>
            <ymin>226</ymin>
            <xmax>331</xmax>
            <ymax>298</ymax>
        </bndbox>
```

```
    </object>
    <object>
        <name>fire</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>362</xmin>
            <ymin>222</ymin>
            <xmax>733</xmax>
            <ymax>280</ymax>
        </bndbox>
    </object>
</annotation>
```

C. Preprocessing

Pada tahapan *preprocessing* bertujuan untuk mempersiapkan data-data sebelum dilakukan proses Training Model. Beberapa tahap yang dilakukan diantaranya terdapat *Image to Array*, *Resize* dan *Augmentasi*.

1. *Image to Array*

Image to array langkah umum dalam *preprocessing* gambar saat bekerja dengan pembelajaran mesin atau tugas visi komputer. Proses ini melibatkan konversi gambar menjadi representasi numerik yang dapat dimasukkan ke dalam model.

2. *Resize Image*

Resize image merupakan tahap yang bertujuan untuk memperkecil jumlah *pixel* daripada *image* hasil penangkapan gambar atau citra digital. Pada tahap ini

penulis memperkecil ukuran gambar dari 800 x 600 pixel menjadi sebesar 256 x 256 pixel yang dimana dapat dilihat pada program dibawah ini

```
def convert_image_to_array(image_dir):

try:

image = cv2.imread(image_dir)

if image is not None :

image = cv2.resize(image, default_image_size)

return img_to_array(image)

else :

return np.array([])

except Exception as e:

print(f"Error : {e}")

return None
```

3. Augmentasi

Augmentasi merupakan proses memodifikasi atau memanipulasi suatu citra, sehingga citra asli dalam bentuk standar akan diubah bentuk dan posisinya.

Augmentasi data bertujuan agar mesin dapat belajar dan mengenali dari berbagai citra yang berbeda-beda sekaligus bisa dimanfaatkan untuk memperbanyak data, dapat dilihat pada program dibawah ini .

```
aug = ImageDataGenerator(
    rotation_range=25,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)
```

Parameter *rotation_range* pada program diatas merupakan parameter yang dapat memutar gambar secara acak antara 0 sampai 25 derajat dengan memberikan nilai bilangan bulat. Saat gambar diputar, beberapa *pixel* akan bergerak keluar gambar dan membuat area kosong yang harus diisi dengan menetapkan parameter *fill_mode* dengan nilai defaultnya adalah *nearest* yang hanya menggantikan area kosong dengan nilai *pixel* terdekat.

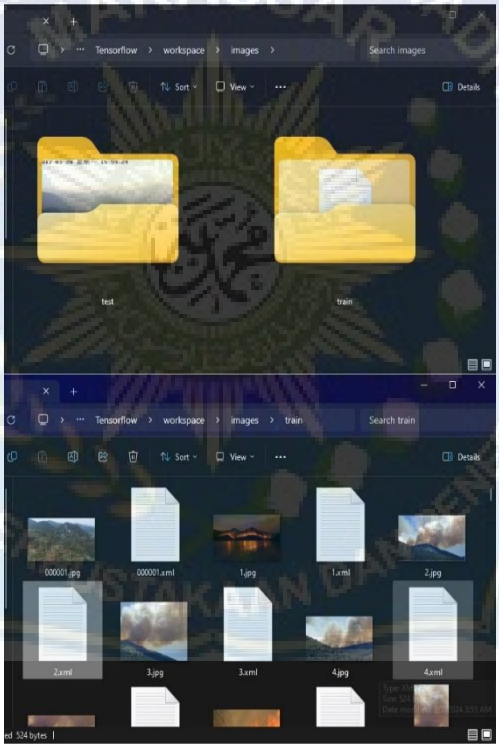
Objek tidak selalu berada di tengah gambar. Untuk mengatasi masalah ini, yaitu dengan cara menggeser *pixel* dengan menambah parameter *width_shift_range* untuk pergeseran gambar secara *horizontal* dan *height_shift_range* untuk pergeseran gambar secara *vertical*. Gambar juga dibalik sepanjang sumbu *horizontal* dengan menambah parameter *horizontal_flip* dengan nilai default true, dan gambar diperbesar dengan menambahkan parameter *zoom_range*, serta sudut digeser dengan menambahkan parameter *shear_range*.

4. Pembagian Data

Pada tahap ini dataset dibagi menjadi 2 bagian. Yang pertama, digunakan untuk training dan yang kedua digunakan untuk testing, untuk data traning digunakan sebesar 70% atau 665 gambar dari total data, dan proporsi untuk testing sebanyak 30% atau 285 gambar dari total data.

Pada Gambar 10 merupakan proses pembagian data, pada gambar sebelah kiri merupakan script untuk membagi keseluruhan gambar dan annotasi kedalam folder traning dan testing dengan proporsi 70% untuk traning dan 30% untuk testing.

D. Pembuatan Model *MobileNet*



Gambar 10 *Flowchart* Rancangan Sistem

```
# Gabungkan Fire Smoke (Image dan Anotasi) dan Split menjadi
Train Test Folder
```

```
# Gabungkan ke folder masing2 (anotasi dan Img)
for img in os.listdir('fire_smoke_final/img'):
  shutil.copy('fire_smoke_final/img/' + img, 'all_img/')
for anot in os.listdir('fire_smoke_final/anotasi'):
  shutil.copy('fire_smoke_final/anotasi/' + anot,
'all_anotasi/')


# IMG + Anotasi, pisahkan jadi Train Test Folder
feature_smoke_img = 'all_img'
label_smoke_anot = 'all_anotasi'


X_train, X_test, y_train, y_test =
train_test_split(feature_smoke_img, label_smoke_anot,
test_size=0.2, random_state=42)


# Create Train Data
for x, y in zip(X_train, y_train):
  shutil.copy(x, 'all_img/' + y + '/train')
  shutil.copy(y, 'all_anotasi/' + y + '/train')


# Create Test Data
for x, y in zip(X_test, y_test):
  shutil.copy(x, 'all_img/' + y + '/test')
  shutil.copy(y, 'all_anotasi/' + y + '/test')
```

Sebelum dilakukan proses segmentasi dengan *MobileNet*, terlebih dahulu dilakukan pembuatan Model. Pada tahap ini akan dilakukan proses *Training Model* dimana data yang telah di-*Split* akan dipelajari untuk dapat menghasilkan Model yang dapat mengenali objek yang akan dideteksi pada gambar (*Feature Learning Process*).

Pada proses *Splitting Data*, dimana dari total 180 gambar akan dibagi menjadi *Data Training* dan *Data Testing* dengan proporsi 70% untuk *Data Training* (126 data) dan 30% untuk *Data Testing* (54 data). Dari data tersebut yang nantinya akan melalui proses pembuatan model. Model inilah yang akan di-*save* dan kemudian di-*load* ketika proses Segmentasi gambar.

1. Hyperparameter

Pada tahap ini *hyperparameter* merupakan variable yang bergantung pada penelitian ini, dikarenakan *hyperparameter* adalah parameter yang nilainya digunakan untuk mengontrol proses pembelajaran.

Tabel 3 *Hyperparameter*

No	Hyperparameter	Value
1	Epoch	**200**
2	Batch size	**8**
3	Activation Function	**RELU 6**
4	Learning Rate	**0.08 (initiate)**

Pada tabel diatas dapat kita lihat bahwasanya memiliki daftar hyperparameter beserta nilainya yang akan digunakan dalam proses pelatihan model. Epoch Model akan diperbaharui dan diperbaiki menggunakan dataset yang tersedia sebanyak 200 kali. Batch size dalam setiap iterasi pelatihan, model akan memproses 8 sampel data sekaligus. Dengan menggunakan batch size kecil seperti ini, pelatihan model menjadi lebih efisien karena memungkinkan penggunaan sumber daya komputasi yang optimal dan mengurangi beban pada memori.

Activation Function fungsi aktivasi yang akan digunakan adalah ReLU6. Fungsi ini memberikan non-linearitas pada model, yang penting dalam pembelajaran dari data yang kompleks. ReLU6 membatasi output neuron menjadi rentang antara 0 hingga 6, memungkinkan model untuk mempelajari representasi data yang lebih terperinci. Learning Rate laju pembelajaran awal adalah 0.08.

Berapa besar langkah yang diambil oleh model dalam memperbarui bobotnya berdasarkan gradien yang dihitung. Model dapat mengkonvergensi ke solusi yang optimal dengan cepat.

E. Training Model

Setelah mengatur Hyperparameter, model selanjutnya akan di traning dengan dataset yang sudah di Preprocessing sebelumnya. Pada proses training nantinya model akan melakukan proses *Feature Learning*, yaitu proses mempelajari feature-feature (dalam hal ini pixel pada gambar) untuk dapat kemudian mengenali objek-objek yang telah dianotasi pada gambar (api dan asap).

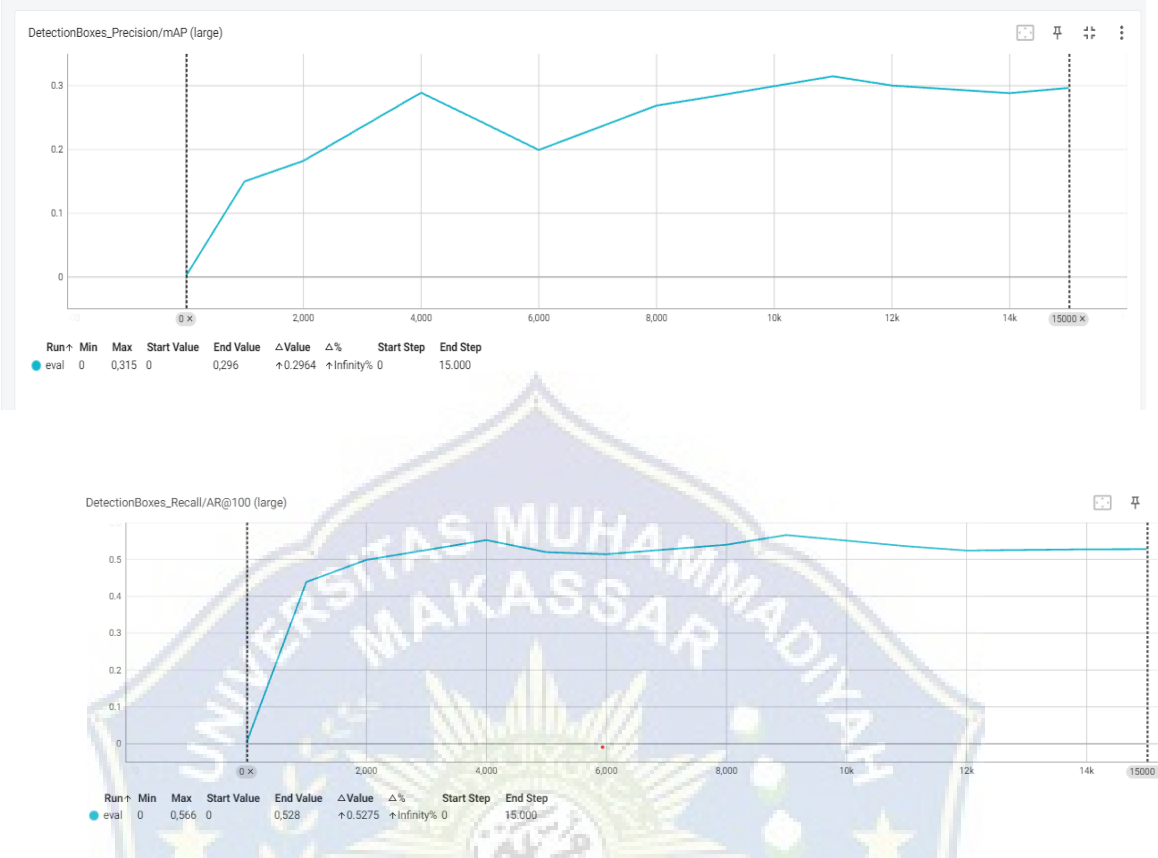
```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection', 'model_main_tf2.py')

command = "python {} --model_dir={} --pipeline_config_path={}
--num_train_steps=15000 --
num_eval_steps=15000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])

!{command}
```

```
python
Tensorflow/models/research/object_detection/model_main_tf2.py
--model_dir=Tensorflow/workspace/models/my_ssd_mobnet --
pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet
/pipeline.config --num_train_steps=15000 --
num_eval_steps=15000
```

Diatas merupakan potongan script untuk proses training model. Nantinya dari script tersebut akan menghasilkan kode yang dapat dijalankan untuk memulai training model. Dapat kita lihat juga pada script tertera jika model akan ditraining sebanyak 20000 step, dimana setiap 1 epoch (proses sekali training) sama dengan 100 step, sehingga totalnya sama dengan 200 epoch (200 kali proses training).

```
C:\WINDOWS\system32\cmd.exe INFO: tensorflow: Step 600 per-
step time 0.193s 10213 09:38:14.490077 28844
model_lib_v2.py:705] Step 600 per-step time 0.193s INFO:
tensorflow: {'Loss/classification_loss': 0.47875477,
'Loss/localization_loss': 0.43792802,
'Loss/regularization_loss': 0.15419038, 'Loss/total_loss':
1.0628732, 'learning_rate': 0.0586664} 10213 09:38:14.490677
28844 model_lib_v2.py:708] {'Loss/classification_loss':
0.47875477, 'Loss/localization_loss': 0.43792802,
'Loss/regularization_loss': 0.15419038, 'Loss/total_loss':
1.0628732, 'learning_rate': 0.0586664} INFO: tensorflow: Step
700 per-step time 0.190s 10213 09:38:33.515778 28844
model_lib_v2.py:705] Step 700 per-step time 0.1905 INFO:
tensorflow: {'Loss/classification_loss': 0.3563194,
'Loss/localization_loss': 0.47075477,
'Loss/regularization_loss': 0.15422885, 'Loss/total_loss':
0.819815, 'learning_rate': 0.0639998} 10213 09:38:33.515770
28844 model_lib_v2.py:708] {'Loss/classification_loss':
0.3563194, 'Loss/localization_loss': 0.47075477,
'Loss/regularization_loss': 0.15422885, 'Loss/total_loss':
0.819815, 'learning_rate': 0.0639998}
```

Source code diatas merupakan proses Training Model, dimana setelah sebelumnya script Training Model dijalankan dan menghasilkan kode untuk memulai proses Training di *Command Prompt*.



Gambar 11 *Mean Average Precision (mAP) dan Average Recall*

Setelah proses Training dilakukan didapatkan hasil pengukuran *Mean Average Precision* dan *Average Recall. Mean Average Precision* menandakan kualitas dan akurasi dari deteksi objek pada berbagai kelas, pada model ini didapatkan nilai *mAP* tertinggi yakni 0,315 atau 31,5%. Lalu untuk *Average Recall* yang memberikan gambaran terhadap seberapa baik model dapat menemukan object pada dataset, didapatkan dengan nilai tertinggi 0,566 tau 56,6%.

F. Evaluasi Model

Setelah proses training model selesai, selanjutnya model akan dievaluasi dengan data testing yang bertujuan untuk mengukur seberapa baik model yang telah dibuat dengan beberapa matriks pengukuran. Dibawah ini adalah potongan script untuk menjalankan proses evaluasi pada model.

```
command = "python {} --model_dir={} --pipeline_config_path={}
--checkpoint_dir={} --
num_eval_steps=15000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'],
paths['CHECKPOINT_PATH'])
```

```
print(command)
```

```
python
Tensorflow/models/research/object_detection/model_main_tf2.py -
-model_dir=Tensorflow/workspace/models/my_ssd_mobnet          --
pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet
/pipeline.config                                              --
checkpoint_dir=Tensorflow/workspace/models/my_ssd_mobnet      --
num_eval_steps=15000
```

```
C:\WINDOWS\system32\cmd. X

INFO: tensorflow: Eval metrics at step 15000

10213 09:46:28.285099 31640 model_lib_v2.py:1015] Eval metrics
at step 15000

INFO:tensorflow: + DetectionBoxes_Precision/mAP: 0.266055

10213    09:46:28.300139    31640    model_lib_v2.py:1018]    +
DetectionBoxes_Precision/mAP: 0.266055
```

INFO: tensorflow: + DetectionBoxes_Precision/mAP@.50Iou: 0.511502

10213 09:46:28.302144 31640 model_lib_v2.py:1018] INFO: tensorflow: + DetectionBoxes_Precision/mAP@.7510U: 0.263107

+ DetectionBoxes_Precision/mAP@.50Iou: 0.511502

+ DetectionBoxes_Precision/mAP@.75I0U: 0.263107 10213 09:46:28.303146 31640 model_lib_v2.py:1018]

INFO: tensorflow: + DetectionBoxes_Precision/mAP (small): 0.023768

10213 09:46:28.304152 31640 model_lib_v2.py:1018] INFO: tensorflow: + DetectionBoxes_Precision/mAP (medium): 0.032407

10213 09:46:28.305154 31640 model_lib_v2.py:1018]

INFO:tensorflow:

+ DetectionBoxes_Precision/mAP (large): 0.288799

10213 09:46:28.306158 31640 model_lib_v2.py:1018] + DetectionBoxes_Precision/mAP (large): 0.288799

INFO : tensorflow: 10213 09:46:28.307165 31640 model_lib_v2.py:1018]

+ DetectionBoxes_Precision/mAP (small): 0.023768

+ DetectionBoxes_Precision/mAP (medium): 0.032407

+ DetectionBoxes_Recall/AR@1: 0.305213

+ DetectionBoxes_Recall/AR@10: 0.407459

+ DetectionBoxes_Recall/AR@100: 0.493027

+ DetectionBoxes_Recall/AR@1: 0.305213

INFO: tensorflow: + DetectionBoxes_Recall/AR@10: 0.407459

10213 + DetectionBoxes_Recall/AR@100:

09:46:28.308174 31640 model_lib_v2.py:1018]

INFO: tensorflow:

0.493027

10213 09:46:28.309176 31640 model_lib_v2.py:1018]

INFO : tensorflow:

+ DetectionBoxes_Recall/AR@100 (small): 0.035294

10213 09:46:28.310179 31640 model_lib_v2.py:1018] +

INFO : tensorflow 10213 09:46:28.311186 31640 model_lib_v2.py:1018] (large): 0.547335

: + DetectionBoxes_Recall/AR@100 (medium): 0.279562 + DetectionBoxes_Recall/AR@100 (medium): 0.279562

INFO:tensorflow:

+ DetectionBoxes_Recall/AR@100 (large): 0.547335 + DetectionBoxes_Recall/AR@100

10213 09:46:28.313192 31640 model_lib_v2.py:1018]

INFO: tensorflow:

+ Loss/localization_loss: 0.413178

DetectionBoxes_Recall/AR@100 (small): 0.035294

+ Loss/localization_loss: 0.413178

+ INFO: tensorflow:

Loss/classification_loss: 0.606683 + Loss/classification_loss: 0.606683

10213 09:46:28.314203 31640 model_lib_v2.py:1018]

INFO: tensorflow: 31640 model_lib_v2.py:1018] + Loss/regularization_loss

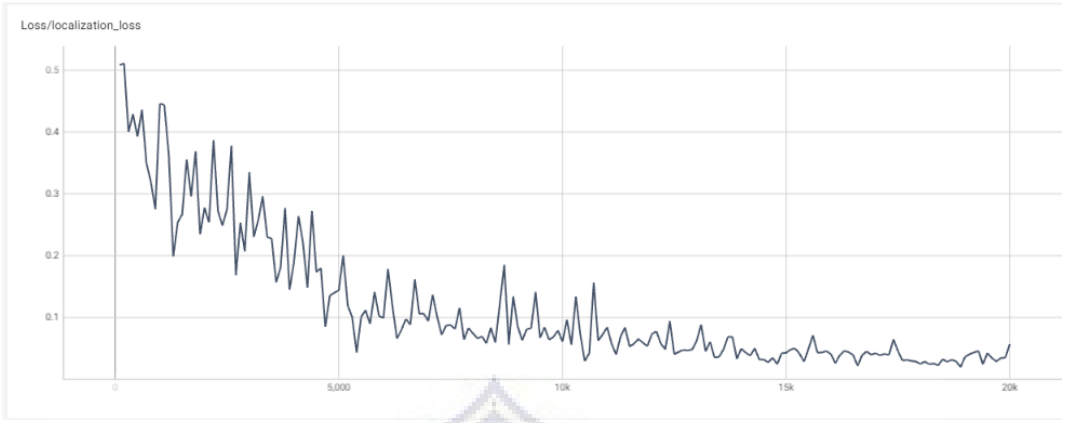
+ Loss/regularization_loss: 0.146187

```
INFO:tensorflow: + Loss/total_loss: 1.166048

10213 09:46:28.315209 + Loss/total_loss: 1.166048

10213 09:46:28.315209 31640 model_lib_v2.py:1018]

: 0.146187

10213 09:46:28.314203 31640 model_lib_v2.py:1018]
```

Dari script tersebut nantinya proses evaluasi akan dijalankan pada Command Prompt seperti pada gambar 15. Setelah script diatas dijalankan, akan dihasilkan dua pengukuran nilai *loss*, yakni

a. *Localization Loss*, mengukur nilai *loss* (kerugian) pada seberapa baik model memprediksi lokasi (kordinat *Bounding Box*) dari objek dalam gambar. Tujuannya adalah agar bounding box yang diprediksi oleh model sesuai dengan *ground truth* bounding box yang sebenarnya. Salah satu metrik yang umum digunakan untuk menghitung localization loss adalah Mean Square Error (MSE) antara prediksi bounding box dan *ground truth* bounding box.
b. *Classification Loss,* mengukur seberapa baik model memprediksi kelas dari objek yang terdapat dalam gambar. Ini berfokus pada aspek klasifikasi, yaitu menentukan jenis objek apa yang terdapat dalam bounding box yang diprediksi oleh model (dalam hal ini api atau asap). Classification Loss biasanya dihitung menggunakan fungsi seperti *Cross Entropy* atau *Binary Cross Entropy*.

Setelah evaluasi dilakukan didapatkan 2 hasil matriks perhitungan nilai eror atau loss, yakni *Localization loss* pada gamabar 17 dan *Classification loss*



Gambar 12 *Localization loss*

Gambar diatas adalah grafik nilai *Localization loss* yang merupakan patokan seberapa baik model dalam memprediksi posisi bounding box pada gambar untuk menentukan posisi objek (api dan asap). Pada grafik sumbu horizontal menunjukan nilai step proses training dan sumbu vertical menunjukan nilai eror/loss.

Berdasarkan grafik, nilai *Localization loss* pada step terakhir (step 20000), yaitu 0.05708 dan nilai *Localization loss* paling rendah terdapat pada step 18900 yaitu 0.01987.

Gambar 13 *Classification loss*

Selanjutnya adalah grafik nilai *Classification loss* yang merupakan patokan seberapa baik model dalam memprediksi jenis objek (api atau asap) yang ada

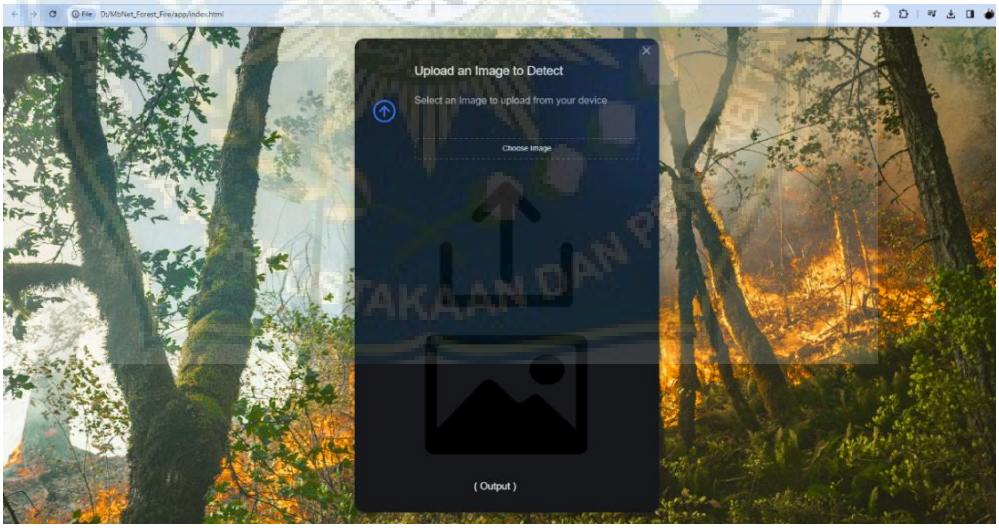
didalam bounding box pada gambar. Pada grafik sumbu horizontal menunjukan nilai step proses training dan sumbu vertical menunjukan nilai eror/loss.

Berdasarkan grafik, nilai *Classification loss* pada step terakhir (step 20000), yaitu 0.1007 dan nilai *Classification loss* paling rendah terdapat pada step 18000 yaitu 0.04903.

G. Perancangan Sistem

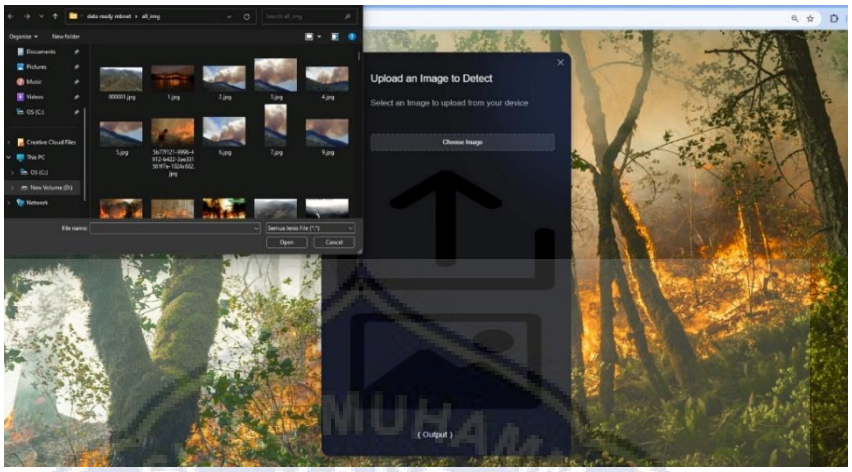
Pada *interface* sistem dibangun menggunakan HTML (*HyperText Markup Language*) sebagai bahasa markah standar, CSS (*Cascading Style Sheets*) sebagai pengatur tampilan elemen,dan JS (*JavaScript*) sebagai pengembangan aplikasi web dan penghubung antara *interface* atau *front end* dan *back end*.

Nantinya data uji coba akan di input gambar dan akan diolah pada *Backend* (Python) melalui API (*Application Programmin g Interface*) dalam bentuk format data JSON (*JavaScript Object Notation*). Setelah kemudian data berhasil diolah oleh Backend untuk mendeteksi api dan asap, output akan kembali diteruskan ke interface web melalui *API*.



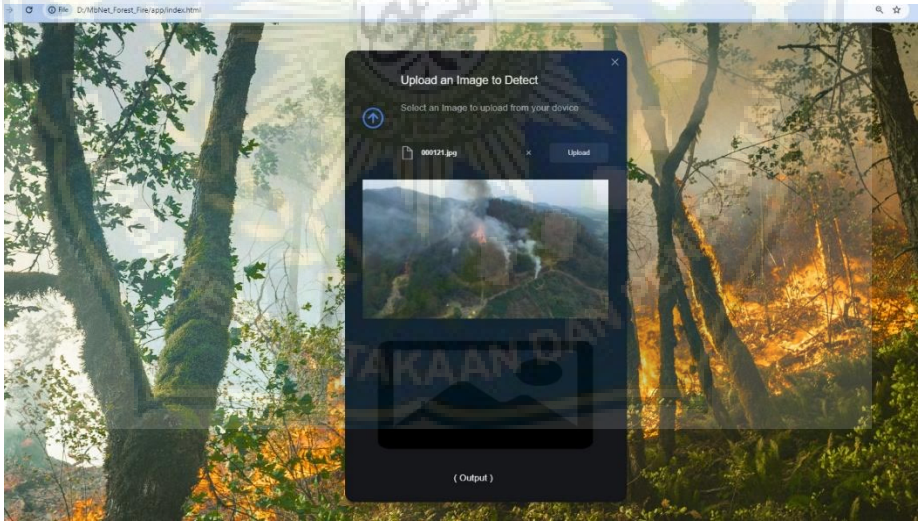
Gambar 14 Tampilan Rancangan *Interface*

Upload gambar ke dalam sistem untuk deteksi proses ini memasukkan file gambar ke dalam aplikasi atau sistem yang dilengkapi dengan model deteksi objek.



Gambar 15 *Upload Image to Detect*

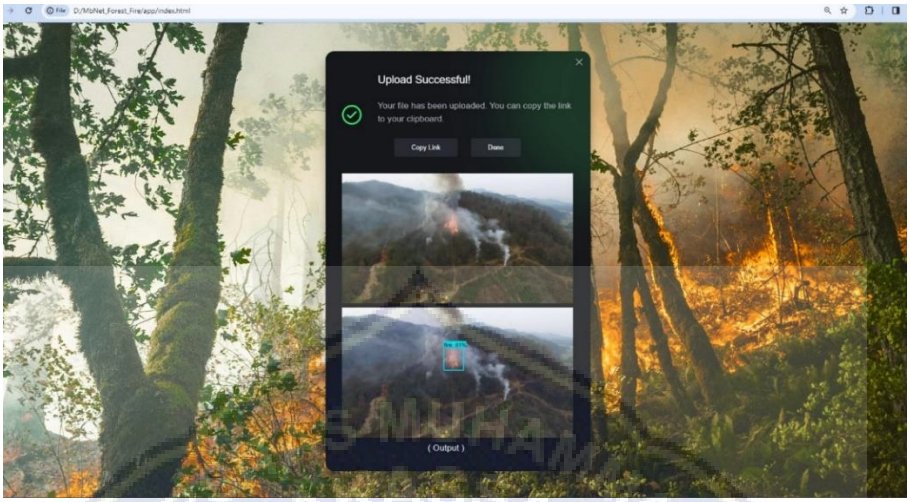
Setelah gambar diunggah, sistem akan menjalankan algoritma deteksi objek untuk mengidentifikasi dan mengklasifikasikan objek-objek yang ada dalam gambar tersebut.



Gambar 16 *Upload Image to Detect*

Setelah proses deteksi selesai, hasilnya dapat berupa gambar yang sama dengan penandaan atau kotak pembatas (*bounding box*) di sekitar objek yang

terdeteksi, atau dapat berupa daftar objek yang terdeteksi beserta informasi tambahan seperti kelas atau label, koordinat, dan skor kepercayaan.



Gambar 17 *Upload Image to Detect*

H. Hasil Output Rancangan

Setelah rancangan sistem untuk deteksi telah dibuat, nantinya data uji coba akan di*upload* dan dikirim ke Backend untuk kemudian dideteksi oleh model MobileNet. Berikut adalah hasil output deteksi kebakaran hutan (gambar 4.13) dan hasil output pada rancangan *interface* (gambar 4.14).

Gambar 18 Hasil *Output* Deteksi

BAB V
KESIMPULAN DAN SARAN

A. Kesimpulan

Dari penelitian yang telah peneliti uraikan, maka dapat ditarik kesimpulan sebagai berikut:

1. Rancangan sistem deteksi api dan asap menggunakan algoritma MobileNet Single Shot Detector (SSD) telah berhasil dikembangkan dan menunjukkan performa yang memadai dalam mendeteksi objek pada citra hutan. Tingkat error yang dihasilkan dari model ini, yaitu 1,987% untuk Localization Loss dan 4,903% untuk Classification Loss, menunjukkan bahwa model memiliki kemampuan yang baik dalam mendeteksi posisi dan jenis objek (api atau asap) secara akurat.
2. Performa model MobileNet SSD dinilai cukup baik dengan nilai Mean Average Precision (mAP) tertinggi sebesar 31,5%, yang menandakan kualitas deteksi objek yang cukup baik. Selain itu, nilai Average Recall tertinggi sebesar 56,6% menunjukkan bahwa model ini efektif dalam menemukan objek yang ada dalam dataset, meskipun masih terdapat ruang untuk peningkatan.

B. Saran

Simulasi ini tentunya masih saja belum sempurna. Masih banyak hal yang dapat dikembangkan pada skripsi ini agar menjadi lebih baik lagi, diantaranya adalah:

1. Untuk meningkatkan akurasi deteksi, disarankan untuk menambah jumlah dan keragaman dataset yang digunakan dalam pelatihan model. Hal ini diharapkan dapat membantu model mengenali objek dengan lebih baik dalam berbagai kondisi.
2. Dalam pengembangan sistem di masa mendatang, disarankan untuk menggunakan sumber daya komputasi yang lebih besar ,hal ini penting untuk memaksimalkan kemampuan model dalam melakukan deteksi.

DAFTAR PUSTAKA

Abror, Z. F. (2019). Klasifikasi Citra Kebakaran Dan Non Kebakaran Menggunakan Convolutional Neural Network. *Jurnal Ilmiah Teknologi Dan Rekayasa*, *24*(2), 102–113. https://doi.org/10.35760/tr.2019.v24i2.2389

Adarrani, A., Putri, W., Susetyo, Y. A., Pada, T., Monitoring, A., Di, S., & Xyz, P. T. (2022). *IMPLEMENTATION OF FLASK FOR STOCK CHECKING IN DISTRIBUTION CENTER & STORE ON MONITORING STOCK APPLICATION IN PT . XYZ IMPLEMENTASI FLASK UNTUK PENGECEKAN STOK DISTRIBUTION CENTER*. *3*(5), 1265–1274.

Adelianthi, N. (2019). Pendeteksi Kebakaran Hutan Menggunakan Komunikasi Lora (Long Range) Wireless Network. *Universitas Islam Negeri Allaudin*, *1*(1), 63. http://www.ghbook.ir/index.php?name=فرهنگ و رسانه های نوین&option=com_dbook&task=readonline&book_id=13650&page=73&chkhashk=ED9C9491B4&Itemid=218&lang=fa&tmpl=component%0Ahttp://www.albayan.ae%0Ahttps://scholar.google.co.id/scholar?hl=en&q=APLIKASI+PENGENA

Alfarizi, M. R. S., Al-farish, M. Z., Taufiqurrahman, M., & ... (2023). Penggunaan Python Sebagai Bahasa Pemrograman untuk Machine Learning dan Deep Learning. *Karimah* https://ojs.unida.ac.id/karimahtauhid/article/view/7518

Amalina, N. (2019). Uji Akurasi Aplikasi Augmented Reality Pembelajaran Huruf Alfabet Bahsa Isyarat Indonesia (BISINDO) pada Vuforia Menggunakan Confusion Matrix. *Universitas Islam Negerimaulana Malik Ibrahim, Malang*.

Andeskob, T. I. (2023). *PROTOTYPE SISTEM PEMANTAUAN DAN PENDETEKSI KEBAKARAN HUTAN DAN LAHAN MENGGUNAKAN TEKNOLOGI WSN BERBASIS IoT*. scholar.unand.ac.id. http://scholar.unand.ac.id/203168/%0Ahttp://scholar.unand.ac.id/203168/10/2. BAB I.pdf

Anggreani, D. (2023). Peningkatan Metode YOLOv7 Dengan Proses Augmentasi Image Pada Klasifikasi Jenis Kupu-Kupu YOLOv7 Method Improvement With Image Augmentation Process In Classification of Butterfly species. *Jtsi*, *4*(2), 243–253.

Ari Kukuh Sentanu, I. G. A., Diafari Djuni, I. G. A. K., & Pramaita, N. (2021). Rancang Bangun Sistem Pendeteksi Kebakaran Hutan Berbasis Node Mcu Esp8266. *Jurnal SPEKTRUM*, *8*(1), 286. https://doi.org/10.24843/spektrum.2021.v08.i01.p32

Ekoputris, R. O. (2018). *MobileNet: Deteksi Objek pada Platform Mobile*. Nodeflux. https://medium.com/nodeflux/mobilenet-deteksi-objek-pada-platform-mobile-bbbf3806e4b3

Irawan, Y., Muzawi, R., Alamsyah, A., Hang Tuah Pekanbaru, U., & Amik Riau, S. (2022). Sistem Real Time Monitoring Pendeteksi Kebakaran Hutan Dan Lahan Di Provinsi Riau Real Time Monitoring System for Forest and Land Fire Detection in Riau Province. *Journal of Information Technology and Computer Science (INTECOMS)*, *5*(2).

Mamuriyah, N., & Sumantri, J. (2022). Penerapan Metode Convolution Neural Network (CNN) Pada Aplikasi Automatic Lip Reading. *Journal of Informatics and Telecommunication Engineering*, *6*(1), 276–287. https://doi.org/10.31289/jite.v6i1.7523

Muharram, R. F., Studi, P., Informatika, T., Gedong, K., Rebo, P., & Timur, J. (2021). *Implementasi Artificial Intelligence Untuk Deteksi*. *01*(03), 1–2.

Nabilah Muhamad. (2023). *Kalimantan Barat Hasilkan Emisi CO2 dari Karhutla Terbanyak sampai Juli 2023 LAYANAN KONSUMEN & KESEHATAN*. Databoks. https://databoks.katadata.co.id/datapublish/2023/08/30/kalimantan-barat-hasilkan-emisi-co2-dari-karhutla-terbanyak-sampai-juli-2023

Ningtyas, D. F., & Setiyawati, N. (2021). Implementasi Flask Framework pada

Pembangunan Aplikasi Purchasing Approval Request. *Jurnal Janitra Informatika Dan Sistem Informasi*, *1*(1), 19–34. https://doi.org/10.25008/janitra.v1i1.120

Nurmalasari, Y. (2019). *Convolutional Neural Network (CNN)*. Medium.

Pranata, A. (2019). *Karhutla di Gunung Bawakaraeng dan Lompobattang Mulai Reda Artikel ini telah tayang di Idntimes.com dengan judul "Karhutla di Gunung Bawakaraeng dan Lompobattang Mulai Reda". Klik untuk baca: https://sulsel.idntimes.com/news/sulsel/aanpranata/karhutla-di-*. IDN TIMES SULSEL. https://sulsel.idntimes.com/news/sulsel/aanpranata/karhutla-di-gunung-bawakaraeng-dan-lompobattang-mulai-reda?page=all

Rahman, S., Sembiring, A., Siregar, D., Khair, H., Gusti Prahmana, I., Puspadini, R., & Zen, M. (2023). Python : Dasar Dan Pemrograman Berorientasi Objek. In *Penerbit Tahta Media*.

Rosaly, R., & Prasetyo, A. (2020). Flowchart Beserta Fungsi dan Simbol-Simbol. *Journal of Chemical Information and Modeling*, *2*(3), 5–7.

Ruldivem, A., Ahmad, U. A., & ... (2022). Desain Dan Implementasi Sistem Pendeteksi Kebakaran Hutan Menggunakan Komunikasi Lora (long Range). *EProceedings* https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/17943

Saputra, R. A., & Faisal Adhinata, dan D. (2023). Model Deteksi Kebakaran Hutan dan Lahan Menggunakan Transfer Learning DenseNet201. *Jurnal of Intelligent System and Computation*, *05*(02), 65–72. https://doi.org/10.52985/insyst.v5i2.317

Syahputra, Z. (2023). Penerapan Ssd-Mobilenet Dalam Identifikasi Jenis Buah Apel. *Indonesian Journal of Education And Computer Science*, *1*(1), 1–7. https://doi.org/10.60076/indotech.v1i1.2

Trivusi. (2022). *Pengertian dan Cara Kerja Algoritma Convolutional Neural*

Network (CNN). Trivusi. https://www.trivusi.web.id/2022/04/algoritma-cnn.html

Wening, T. (2019). *Water Bombing dan Hujan Buatan, Dua Metode untuk Memadamkan Kebakaran Hutan*. Bobo.Id. https://bobo.grid.id/read/081861393/water-bombing-dan-hujan-buatan-dua-metode-untuk-memadamkan-kebakaran-hutan?page=all

Winarno, A., & Awang Joko Mastera. (2023). Desain Sistem Pendeteksi Kebakaran Hutan Dengan Gps Dan Telegram. *TESLA: Jurnal Teknik Elektro*, *25*(1), 1–12. https://doi.org/10.24912/tesla.v25i1.22931

LAMPIRAN

Lampiran 1 Surat Permohonan Penelitian

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
LEMBAGA PENELITIAN PENGEMBANGAN DAN PENGABDIAN KEPADA MASYARAKAT
Jl. Sultan Alauddin No. 259 Telp.866972 Fax (0411)865588 Makassar 90221 e-mail :lp3m@unismuh.ac.id

Nomor : 4465/05/C.4-VIII/VI/1445/2024
Lamp : 1 (satu) Rangkap Proposal
Hal : Permohonan Izin Penelitian

11 June 2024 M
05 Dzulhijjah 1445

Kepada Yth,
Bapak Gubernur Prov. Sul-Sel
Cq. Kepala Dinas Penanaman Modal & PTSP Provinsi Sulawesi Selatan
di –
Makassar

اَلسَّلَامُ عَلَيْكُمْ وَرَحْمَةُ اللهِ وَبَرَكَاتُهُ

Berdasarkan surat Dekan Teknik Universitas Muhammadiyah Makassar, nomor: 377/05/C.4-VI/VI/45/2024 tanggal 10 Juni 2024, menerangkan bahwa mahasiswa tersebut di bawah ini :

Nama : **ABD SALAM**
No. Stambuk : **10584 1110220**
Fakultas : **Teknik**
Jurusan : **Informatika**
Pekerjaan : **Mahasiswa**

Bermaksud melaksanakan penelitian/pengumpulan data dalam rangka penulisan Skripsi dengan judul :

"PENERAPAN ALGORITMA MOBILENET SINGLE SHOT DETECTOR UNTUK DETEKSI API DAN ASAP BERPOTENSI KEBAKARAN PADA CITRA HUTAN"

Yang akan dilaksanakan dari tanggal 14 Juni 2024 s/d 14 Agustus 2024.

Sehubungan dengan maksud di atas, kiranya Mahasiswa tersebut diberikan izin untuk melakukan penelitian sesuai ketentuan yang berlaku.
Demikian, atas perhatian dan kerjasamanya diucapkan Jazakumullahu khaeran

اَلسَّلَامُ عَلَيْكُمْ وَرَحْمَةُ اللهِ وَبَرَكَاتُهُ

Ketua LP3M,

Dr. Muh. Arief Muhsin, M.Pd.
NBM 1127761

06-24

Lampiran 2 Surat Izin Penelitian LP3M

UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

بسم الله الرحمن الرحيم

Nomor : 377/05/C.4-VI/VI/45/2024
Lamp. : -
Hal : **Pengantar Penelitian**

Makassar, 03 Dzulhijjah 1445 H / 10 Juni 2024 M

Kepada yang Terhormat,
Ketua LP3M Unismuh Makassar
Di –
Tempat

Assalamu 'Alaikum Warahmatullahi Wabarakatuh

Dengan Rahmat Allah SWT, Semoga aktivitas kita bernilai ibadah di Sisi – Nya. Dalam rangka penyelesaian Tugas Sarjana / Tugas Akhir Mahasiswa pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar dengan judul: ***"Penerapan Algoritma Mobilenet Single Shot Detector untuk Deteksi Api dan Asap Berpotensi Kebakaran Pada Citra Hutan"***, Sehubungan hal tersebut, maka kami meminta kesedian Bapak/Ibu agar kiranya berkenan membantu perihal surat tersebut. Bersama ini kami sampaikan mahasiswa(i):

No.	Stambuk	Nama
1.	105 84 11102 20	Abd. Salam

Demikian surat kami atas perhatian dan kerja samanya kami haturkan banyak terima kasih.

Jazakumullah Khaeran Katsiran
Wassalamu 'Alaikum warahmatullah Wabarakatuh

Ketua Program Studi
Informatika

Muhyiddin M. Hayat, S.Kom., MT.
NBM. 150 4577

Tembusan: Kepada Yang Terhormat,
1 Dekan Fakultas Teknik
2 Arsip

Gedung Menara Iqra Lantai 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Web: https://teknik.unismuh.ac.id/, e-mail: teknik@unismuh.ac.id

PEMERINTAH PROVINSI SULAWESI SELATAN
DINAS PENANAMAN MODAL DAN PELAYANAN TERPADU SATU PINTU
Jl.Bougenville No.5 Telp. (0411) 441077 Fax. (0411) 448936
Website : http://simap-new.sulselprov.go.id Email : ptsp@sulselprov.go.id
Makassar 90231

Nomor : **15230/S.01/PTSP/2024**
Lampiran : -
Perihal : **Izin penelitian**

Kepada Yth.
Bupati Gowa

di-
Tempat

Berdasarkan surat Ketua LP3M UNISMUH Makassar Nomor : 4465/05/C.4-VIII/V/1445/2024 tanggal 11 Juni 2024 perihal tersebut diatas, mahasiswa/peneliti dibawah ini:

N a m a	: **ABD SALAM**
Nomor Pokok	: 105841110220
Program Studi	: Informatika
Pekerjaan/Lembaga	: Mahasiswa (S1)
Alamat	: Jl. Sit Alauddin, No. 259 Makassar

PROVINSI SULAWESI SELATAN

Bermaksud untuk melakukan penelitian di daerah/kantor saudara dalam rangka menyusun SKRIPSI, dengan judul :

" PENERAPAN ALGORITMA MOBILENET SINGLE SHOT DETECTOR UNTUK DETEKSI API DAN ASAP BERPOTENSI KEBAKARAN PADA CITRA HUTAN "

Yang akan dilaksanakan dari : Tgl. ***14 Juni s/d 14 Juli 2024***

Sehubungan dengan hal tersebut diatas, pada prinsipnya kami ***menyetujui*** kegiatan dimaksud dengan ketentuan yang tertera di belakang surat izin penelitian.

Demikian Surat Keterangan ini diberikan agar dipergunakan sebagaimana mestinya.

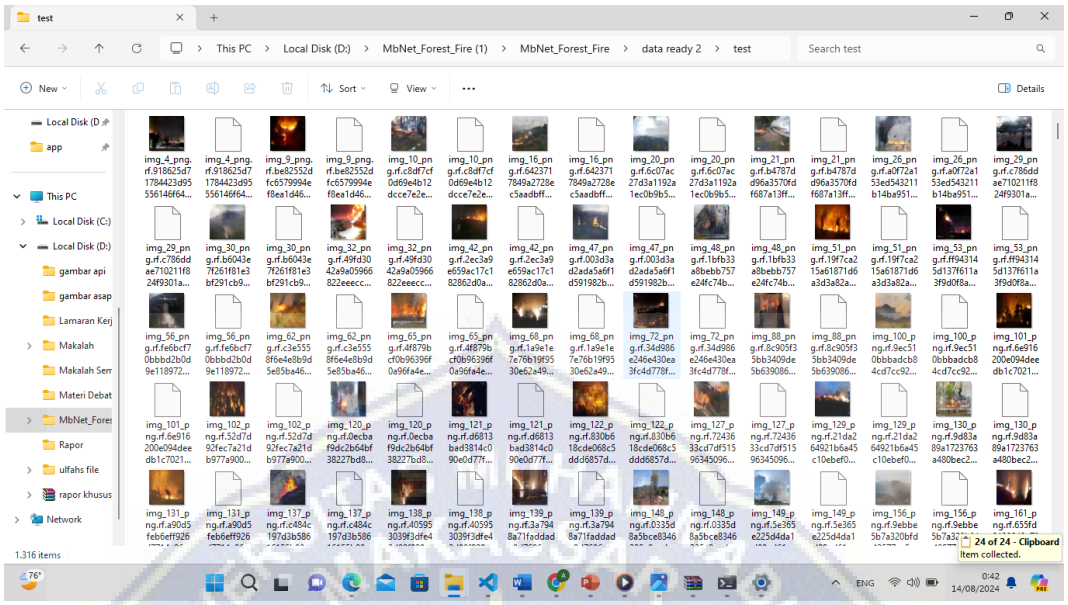
Diterbitkan di Makassar
Pada Tanggal 11 Juni 2024

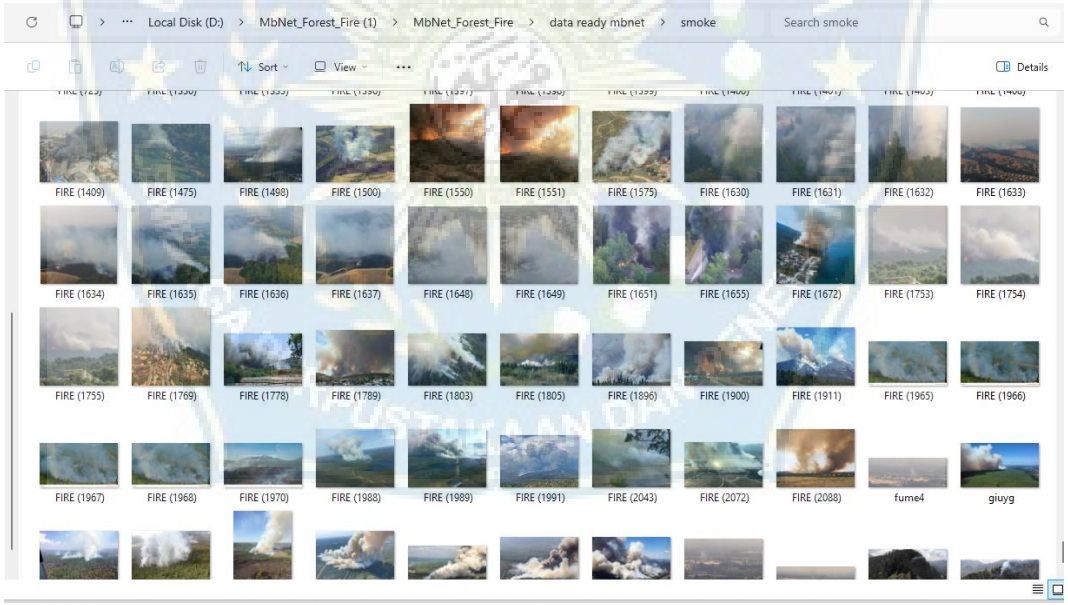
KEPALA DINAS PENANAMAN MODAL DAN PELAYANAN TERPADU SATU PINTU PROVINSI SULAWESI SELATAN

ASRUL SANI, S.H., M.Si.
Pangkat : PEMBINA TINGKAT I
Nip : 19750321 200312 1 008

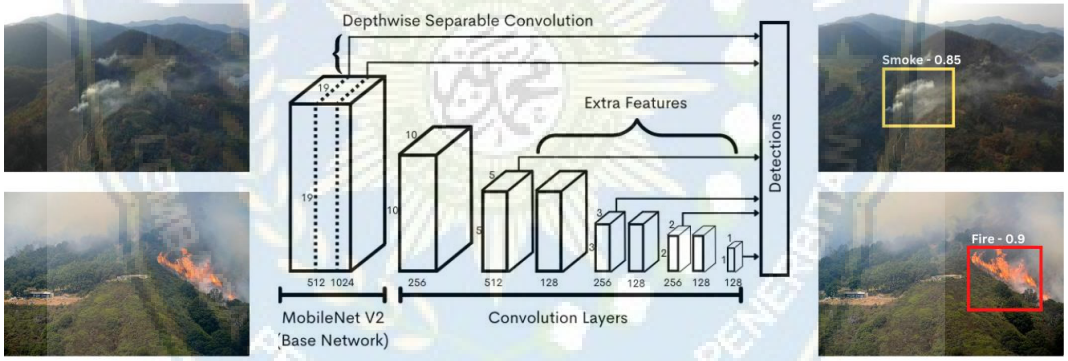
Tembusan Yth
1. Ketua LP3M UNISMUH Makassar di Makassar;
2. *Pertinggal.*

Lampiran 3 *Data Mentah*





Lampiran 4 Data Hasil *Cluster*



Lampiran 5 *Source Code*

```
!pip install tensorflow==2.17.0
```

```
import os

CUSTOM_MODEL_NAME = 'my_custom_model'  # Pastikan untuk
mendefinisikan ini sebelum menggunakannya

paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace',
'annotations'),
    'MODEL_PATH': os.path.join('Tensorflow',
'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow',
'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow', 'protoc')
}


CUSTOM_MODEL_NAME = 'my_custom_model'  # Pastikan untuk
mendefinisikan ini sebelum menggunakannya

paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
```

```
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace',
'annotations'),
    'MODEL_PATH': os.path.join('Tensorflow',
'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow',
'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow', 'protoc')
}
```

```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME =
'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20
200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

```
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow',
'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow',
'workspace','images'),
```

```
    'MODEL_PATH': os.path.join('Tensorflow',
'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow',
'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }
```

```
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'],
TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'],
LABEL_MAP_NAME)
}
```

```
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

```
# https://www.tensorflow.org/install/source_windows
```

```
if os.name=='nt':
```

```
    !pip install wget
    import wget
```

```
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection')):
    !git clone https://github.com/tensorflow/models
{paths['APIMODEL_PATH']}
```

```
# Install Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc
object_detection/protos/*.proto --python_out=. && cp
object_detection/packages/tf2/setup.py . && python -m pip
install .

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/
download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-
win64.zip
    os.environ['PATH'] += os.pathsep +
os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc
object_detection/protos/*.proto --python_out=. && copy
object_detection\\packages\\tf2\\setup.py setup.py && python
setup.py build && python setup.py install
    !cd Tensorflow/models/research/slim && pip install -e .
```

```
!pip list
```

```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection', 'builders',
'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}
```

```
import tensorflow as tf
```

```
tf.config.list_physical_devices('GPU')
```

```
print("Num GPUs Available: ",
len(tf.config.list_physical_devices('GPU')))
```

```
# !pip install tensorflow==2.4.1 tensorflow-gpu==2.4.1 --
upgrade
```

```
# !pip uninstall protobuf matplotlib -y
# !pip install protobuf matplotlib==3.2
```

```
!pip list
```

```
import object_detection
```

```
if os.name =='posix':
    !wget {PRETRAINED_MODEL_URL}
    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'}
{paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf
{PRETRAINED_MODEL_NAME+'.tar.gz'}
if os.name == 'nt':
    wget.download(PRETRAINED_MODEL_URL)
```

```
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'}
{paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf
{PRETRAINED_MODEL_NAME+'.tar.gz'}
```

```
labels = [{'name':'smoke', 'id':1}, {'name':'fire', 'id':2}]

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

```
# OPTIONAL IF RUNNING ON COLAB
ARCHIVE_FILES = os.path.join(paths['IMAGE_PATH'],
'archive.tar.gz')
if os.path.exists(ARCHIVE_FILES):
  !tar -zxvf {ARCHIVE_FILES}
```

```
if not os.path.exists(files['TF_RECORD_SCRIPT']):
    !git clone
https://github.com/nicknochnack/GenerateTFRecord
{paths['SCRIPTS_PATH']}
```

```
!pip install pytz
```

```
!python {files['TF_RECORD_SCRIPT']} -x
{os.path.join(paths['IMAGE_PATH'], 'train')} -l
{files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'],
'train.record')}
!python {files['TF_RECORD_SCRIPT']} -x
{os.path.join(paths['IMAGE_PATH'], 'test')} -l
```

```
{files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'],
'test.record')}
```

```
if os.name =='posix':
    !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'],
PRETRAINED_MODEL_NAME, 'pipeline.config')}
{os.path.join(paths['CHECKPOINT_PATH'])}
if os.name == 'nt':
    !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'],
PRETRAINED_MODEL_NAME, 'pipeline.config')}
{os.path.join(paths['CHECKPOINT_PATH'])}
```

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

```
config =
config_util.get_configs_from_pipeline_file(files['PIPELINE_CON
FIG'])
```

```
config
```

```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
```

```
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 8
```

```
pipeline_config.train_config.fine_tune_checkpoint =
os.path.join(paths['PRETRAINED_MODEL_PATH'],
PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type =
"detection"
pipeline_config.train_input_reader.label_map_path=
files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.inpu
t_path[:] = [os.path.join(paths['ANNOTATION_PATH'],
'train.record')]
pipeline_config.eval_input_reader[0].label_map_path =
files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.in
put_path[:] = [os.path.join(paths['ANNOTATION_PATH'],
'test.record')]
```

```
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection', 'model_main_tf2.py')
```

```
command = "python {} --model_dir={} --pipeline_config_path={}
--num_train_steps=15000 --
num_eval_steps=15000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
```

```
print(command)
```

```
!{command}
```

```
command = "python {} --model_dir={} --pipeline_config_path={}
--checkpoint_dir={} --
num_eval_steps=15000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'],
paths['CHECKPOINT_PATH'])
```

```
print(command)
```

```
!{command}
```

```
!pip install tf
```

```
!pip install --upgrade tensorflow tensorflow-hub tensorflow-
models-official
```

```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as
viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
```

```
# # Prevent GPU complete consumption
# gpus = tf.config.list_physical_devices('GPU')
# if gpus:
#     try:
```

```
#         tf.config.experimental.set_virtual_device_configurat
ion(
#             gpus[0],
[tf.config.experimental.VirtualDeviceConfiguration(memory_limi
t=5120)])
#     except RunTimeError as e:
#         print(e)
```

```
# Load pipeline config and build a detection model
configs =
config_util.get_configs_from_pipeline_file(files['PIPELINE_CON
FIG'])
detection_model =
model_builder.build(model_config=configs['model'],
is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-
16')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict,
shapes)
    return detections
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

```
category_index =
label_map_util.create_category_index_from_labelmap(files['LABE
LMAP'])
```

```
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'testimg2.jpg')
```

```
imgp = r'Tensorflow\workspace\images\test\000118.jpg'
```

```
img = cv2.imread(imgp)
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np,
0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
```

```
            max_boxes_to_draw=5,
            min_score_thresh=.3,
            agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections,
cv2.COLOR_BGR2RGB))
plt.show()
```

```
detections.keys()
```

```
!pip install easyocr
```

```
!pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111
torchaudio===0.8.1 -f
https://download.pytorch.org/whl/torch_stable.html
```

```
import easyocr
```

```
detection_threshold = 0.7
```

```
image = image_np_with_detections
scores = list(filter(lambda x: x> detection_threshold,
detections['detection_scores']))
boxes = detections['detection_boxes'][:len(scores)]
classes = detections['detection_classes'][:len(scores)]
```

```
width = image.shape[1]
height = image.shape[0]
```

```
# Apply ROI filtering and OCR
for idx, box in enumerate(boxes):
```

```
    print(box)
    roi = box*[height, width, height, width]
    print(roi)
    region =
image[int(roi[0]):int(roi[2]),int(roi[1]):int(roi[3])]
    reader = easyocr.Reader(['en'])
    ocr_result = reader.readtext(region)
    print(ocr_result)
    plt.imshow(cv2.cvtColor(region, cv2.COLOR_BGR2RGB))
```

```
for result in ocr_result:
    print(np.sum(np.subtract(result[0][2],result[0][1])))
    print(result[1])
```

```
region_threshold = 0.05
```

```
def filter_text(region, ocr_result, region_threshold):
    rectangle_size = region.shape[0]*region.shape[1]

    plate = []
    for result in ocr_result:
        length = np.sum(np.subtract(result[0][1],
result[0][0]))
        height = np.sum(np.subtract(result[0][2],
result[0][1]))

        if length*height / rectangle_size > region_threshold:
            plate.append(result[1])
    return plate
```

```
filter_text(region, ocr_result, region_threshold)
```

```
region_threshold = 0.6
```

```
def ocr_it(image, detections, detection_threshold,
region_threshold):

    # Scores, boxes and classes above threhold
    scores = list(filter(lambda x: x> detection_threshold,
detections['detection_scores']))
    boxes = detections['detection_boxes'][:len(scores)]
    classes = detections['detection_classes'][:len(scores)]

    # Full image dimensions
    width = image.shape[1]
    height = image.shape[0]

    # Apply ROI filtering and OCR
    for idx, box in enumerate(boxes):
        roi = box*[height, width, height, width]
        region =
image[int(roi[0]):int(roi[2]),int(roi[1]):int(roi[3])]
        reader = easyocr.Reader(['en'])
        ocr_result = reader.readtext(region)

        text = filter_text(region, ocr_result,
region_threshold)

        plt.imshow(cv2.cvtColor(region, cv2.COLOR_BGR2RGB))
        plt.show()
        print(text)
        return text, region
text, region = ocr_it(image_np_with_detections, detections,
detection_threshold, region_threshold)
```

```
import csv
import uuid
```

```
'{}.jpg'.format(uuid.uuid1())
```

```
def save_results(text, region, csv_filename, folder_path):
    img_name = '{}.jpg'.format(uuid.uuid1())

    cv2.imwrite(os.path.join(folder_path, img_name), region)

    with open(csv_filename, mode='a', newline='') as f:
        csv_writer = csv.writer(f, delimiter=',',
quotechar='"', quoting=csv.QUOTE_MINIMAL)
        csv_writer.writerow([img_name, text])
```

```
region
```

```
save_results(text, region, 'detection_results.csv',
'Detection_Images')
```

```
!pip uninstall opencv-python-headless -y
```

```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor =
tf.convert_to_tensor(np.expand_dims(image_np, 0),
dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
```

```
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offse
t,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=5,
                min_score_thresh=.5,
                agnostic_mode=False)

    try:
        text, region = ocr_it(image_np_with_detections,
detections, detection_threshold, region_threshold)
        save_results(text, region, 'realtimeresults.csv',
'Detection_Images')
    except:
        pass

    cv2.imshow('object
detection',  cv2.resize(image_np_with_detections, (800, 600)))

    if cv2.waitKey(10) & 0xFF == ord('q'):
        cap.release()
```

```
        cv2.destroyAllWindows()
        break
FREEZE_SCRIPT = os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection', 'exporter_main_v2.py ')
```

```
command = "python {} --input_type=image_tensor --
pipeline_config_path={} --trained_checkpoint_dir={} --
output_directory={}".format(FREEZE_SCRIPT
,files['PIPELINE_CONFIG'], paths['CHECKPOINT_PATH'],
paths['OUTPUT_PATH'])
```

```
print(command)
```

```
!{command}
```

```
!pip install tensorflowjs
```

```
command = "tensorflowjs_converter --
input_format=tf_saved_model --
output_node_names='detection_boxes,detection_classes,detection
_features,detection_multiclass_scores,detection_scores,num_det
ections,raw_detection_boxes,raw_detection_scores' --
output_format=tfjs_graph_model --
signature_name=serving_default {}
{}".format(os.path.join(paths['OUTPUT_PATH'], 'saved_model'),
paths['TFJS_PATH'])
```

```
print(command)
```

```
!{command}
```

```
# Test Code:
https://github.com/nicknochnack/RealTimeSignLanguageDetectionw
ithTFJS
```

```
TFLITE_SCRIPT = os.path.join(paths['APIMODEL_PATH'],
'research', 'object_detection', 'export_tflite_graph_tf2.py ')
```

```
command = "python {} --pipeline_config_path={} --
trained_checkpoint_dir={} --
output_directory={}".format(TFLITE_SCRIPT
,files['PIPELINE_CONFIG'], paths['CHECKPOINT_PATH'],
paths['TFLITE_PATH'])
print(command)
```

```
!{command}
```

```
FROZEN_TFLITE_PATH = os.path.join(paths['TFLITE_PATH'],
'saved_model')
TFLITE_MODEL = os.path.join(paths['TFLITE_PATH'],
'saved_model', 'detect.tflite')
```

```
command = "tflite_convert \
--saved_model_dir={} \
--output_file={} \
--input_shapes=1,300,300,3 \
--input_arrays=normalized_input_image_tensor \
--
output_arrays='TFLite_Detection_PostProcess','TFLite_Detection
_PostProcess:1','TFLite_Detection_PostProcess:2','TFLite_Detec
tion_PostProcess:3' \
--inference_type=FLOAT \
--allow_custom_ops".format(FROZEN_TFLITE_PATH, TFLITE_MODEL, )
```

```
print(command)
```

```
!{command}
```

```
!tar -czf models.tar.gz {paths['CHECKPOINT_PATH']}
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Lampiran 6 Hasil Validasi

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
UPT PERPUSTAKAAN DAN PENERBITAN
Alamat kantor: Jl.Sultan Alauddin N0.259 Makassar 90221 Tlp.(0411) 866972,881593, Fax.(0411) 865588

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

SURAT KETERANGAN BEBAS PLAGIAT

UPT Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar,
Menerangkan bahwa mahasiswa yang tersebut namanya di bawah ini:

Nama : Arvianda

Nim : 105841102520

Program Studi : Teknik Informatika

Dengan nilai:

No	Bab	Nilai	Ambang Batas
1	Bab 1	6 %	10 %
2	Bab 2	22 %	25 %
3	Bab 3	8 %	10 %
4	Bab 4	3 %	10 %
5	Bab 5	4 %	5 %

Dinyatakan telah lulus cek plagiat yang diadakan oleh UPT- Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar Menggunakan Aplikasi Turnitin.

Demikian surat keterangan ini diberikan kepada yang bersangkutan untuk dipergunakan seperlunya.

Makassar, 14 Agustus 2024
Mengetahui,

Kepala UPT- Perpustakaan dan Pernerbitan,

Nursinah, S.Kom.,M.I.P
NBM. 964 591

Jl. Sultan Alauddin no 259 makassar 90222
Telepon (0411)866972,881 593,fax (0411)865 588
Website: www.library.unismuh.ac.id
E-mail : perpustakaan@unismuh.ac.id