New Microsoft Word Document.pdf

anonymous marking enabled

Submission date: 22-Jan-2025 12:31PM (UTC-0600)
Submission ID: 2569035148
File name: New_Microsoft_Word_Document.pdf (574.15K)
Word count: 3556
Character count: 20642

Road Damage Detection Using Tensorflow Object Detection API and Tensorflow js
[Deteksi Kerusakan Jalan Menggunakan Tensorflow Object Detection API dan Tensorflow js]

Ahmad Irfan Masaid[1)], Suhendro Busono [*,2)]
[1), 2)] Program Studi Teknik Informatika, Universitas Muhammadiyah Sidoarjo, Indonesia
*Email Penulis Korespondensi: hendrob@umsida.ac.id

Abstract. *Road damage significantly impacts mobility and the economy, with conventional detection methods being time-consuming and costly. This research develops an automated road damage detection system using the Single Shot Detector (SSD) with the MobileNetV2 architecture and TensorFlow Object Detection API, implemented in a web application. The system detects three types of road damage on asphalt and concrete surfaces using a 2019 dataset. Testing involved varying parameters such as learning rate, batch size, data split and total training steps. The best performance was achieved for augmented data model with a learning rate of 0,01 and batch size of 32, yielding a precision of 87.4%, recall of 84.8%, and F1-score of 83.8%. The implementation on a web application facilitates real-time road damage detection with improved accessibility.*

Keywords *– deep learning; road damage; SSD Mobilenet; Tensorflow; TensorflowJS*

Abstrak. *Kerusakan jalan berdampak signifikan terhadap mobilitas dan perekonomian, dengan metode deteksi konvensional yang memakan waktu dan biaya tinggi. Penelitian ini mengembangkan sistem deteksi kerusakan jalan otomatis menggunakan Single Shot Detector (SSD) dengan arsitektur MobileNetV2 dan TensorFlow Object Detection API, yang diterapkan dalam aplikasi web. Sistem ini mendeteksi tiga jenis kerusakan jalan pada permukaan aspal dan beton menggunakan dataset tahun 2019. Pengujian melibatkan variasi parameter seperti laju pembelajaran, ukuran batch, dan total langkah pelatihan. Kinerja terbaik dicapai pada model dengan augmentasi data dan konfigurasi learning rate 0,01, dengan hasil precision 87,4%, recall 84,8%, dan F1-score 83,8%. Implementasi pada aplikasi web memungkinkan deteksi kerusakan jalan secara real-time dengan peningkatan aksesibilitas.*

Kata Kunci *– deep learning; kerusakan jalan; SSD Mobilenet; Tensorflow; TensorflowJS*

I. Pendahuluan

Kondisi infrastruktur jalan yang baik merupakan aspek penting dalam mendukung mobilitas dan pertumbuhan ekonomi suatu daerah. Namun, kerusakan jalan masih menjadi permasalahan serius di banyak wilayah, yang dapat menyebabkan ketidaknyamanan pengguna jalan, meningkatkan risiko kecelakaan, dan menimbulkan kerugian ekonomi [1]. Penelitian ini bertujuan untuk menciptakan sistem deteksi kerusakan jalan yang lebih responsif, untuk mengatasi permasalahan keterlambatan identifikasi kerusakan jalan yang dapat meningkatkan risiko kecelakaan dan kerugian ekonomi. Metode konvensional untuk mendeteksi dan memantau kerusakan jalan seringkali membutuhkan waktu yang lama, tenaga kerja yang intensif, dan biaya yang tinggi [2]. Hal ini menyebabkan keterlambatan dalam identifikasi dan perbaikan kerusakan jalan, yang pada akhirnya dapat memperburuk kondisi infrastruktur secara keseluruhan.

Berkembang pesatnya teknologi kecerdasan buatan (Artifical Intellegence) dan computer vision, membuka peluang untuk mengembangkan sistem deteksi kerusakan jalan yang lebih efisien dan akurat. Salah satu pendekatan yang menjanjikan adalah penggunaan algoritma deep learning untuk deteksi objek, seperti Single Shot Detector (SSD) dengan arsitektur MobileNetV2. SSD MobileNetV2 dipilih karena kemampuannya dalam melakukan deteksi objek dengan akurasi tinggi namun tetap efisien dalam penggunaan sumber daya komputasi. Hal ini memungkinkan implementasi model pada berbagai jenis perangkat, termasuk perangkat mobile dan sistem embedded.

TensorFlow, sebagai salah satu framework machine learning yang populer, menyediakan tools dan library yang kuat untuk mengembangkan dan melatih model SSD MobileNetV2. Khususnya, TensorFlow Object Detection API menawarkan framework yang fleksibel untuk pengembangan model deteksi objek. API ini menyediakan berbagai arsitektur model pre-trained, termasuk SSD MobileNetV2, yang dapat digunakan untuk transfer learning [3]. Hal ini memungkinkan pengembang untuk memanfaatkan model yang telah dilatih pada dataset besar untuk tugas-tugas spesifik dengan dataset yang lebih kecil.

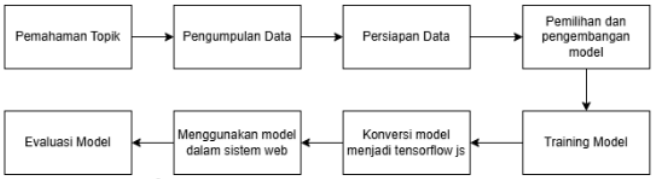
Sementara itu, TensorFlow.js memungkinkan deployment model ke dalam lingkungan web, membuka peluang untuk pengembangan aplikasi berbasis browser yang dapat diakses secara luas. Dengan memanfaatkan kombinasi TensorFlow Object Detection API dan TensorFlow.js, penelitian ini dapat fokus pada optimisasi model untuk kasus spesifik deteksi kerusakan jalan, tanpa perlu membangun seluruh pipeline deteksi objek dari awal [4]. Penelitian ini

bertujuan untuk mengembangkan sistem deteksi kerusakan jalan otomatis menggunakan SSD MobileNetV2 dengan TensorFlow, serta mengimplementasikannya dalam aplikasi web menggunakan TensorFlow.js.

Dengan menggabungkan kekuatan machine learning, deep learning, computer vision, dan teknologi web modern, penelitian ini berpotensi memberikan solusi inovatif untuk masalah pengelolaan infrastruktur jalan yang telah lama ada, sambil juga membuka jalan bagi pengembangan aplikasi Artificial Intellegence yang lebih luas di bidang manajemen infrastruktur perkotaan

II. Metode

Penelitian ini mengikuti beberapa tahapan untuk dapat melatih model yang dapat mendeteksi kerusakan jalan, Adapun langkah-langkah dalam membangun sistem ini adalah sebagai berikut:



Gambar 1. Urutan langkah penelitan

A. Perencanaan

Pada tahap ini, peneliti akan melakukan berbagai kegiatan untuk memahami topik penelitian, merumuskan tujuan dan ruang lingkup penelitian, mengidentifikasi data yang diperlukan, merancang sistem, dan membuat rencana penelitian. Kegiatan pemahaman topik penelitian bertujuan untuk memberikan pemahaman yang mendalam tentang topik penelitian, termasuk konsep-konsep dasar, teori-teori yang relevan, dan penelitian-penelitian terdahulu yang terkait.

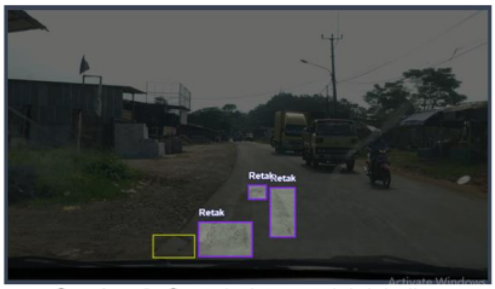
B. Pengumpulan data

Untuk mengembangkan sistem deteksi kerusakan jalan yang efektif, langkah pertama adalah melakukan pengumpulan data. Dalam penelitian ini, digunakan dataset publik dari *Kaggle* [5] yang mencakup berbagai gambar kerusakan jalan. Dataset yang digunakan dalam penelitian ini sebanyak 1018 gambar kerusakan jalan.

Gambar 2. Sampel dataset yang digunakan

C. Persiapan data

Pada tahap ini, peneliti akan melakukan labeling, preprocessing dan augmentasi data untuk meningkatkan variasi dataset dan mempersiapkan data untuk pelatihan model, proses tersebut dilakukan di platform Roboflow. Tiga kategori kerusakan jalan yang akan dideteksi dalam penelitian ini: Retak, Lubang dan Tambalan.



Gambar 3. Contoh data setelah labeling

Untuk meningkatkan variasi dataset dan membantu model belajar lebih baik, dilakukan augmentasi data. Teknik augmentasi yang digunakan meliputi flipping horizontal dan vertikal, serta rotasi 90° searah jarum jam, berlawanan arah jarum jam, dan terbalik.

Gambar 4. Contoh data setelah proses augmentasi

Dataset yang sebelumnya berjumlah 1018 data, bertambah menjadi 1600 data setelah dilakukan proses augmentasi.

D. Pengembangan dan pelatihan model

Pada tahap ini, dilakukan pemilihan arsitektur model, mengimplementasikan model menggunakan Tensorflow Object Detection API dan melakukan transfer learning dan juga fine tunning pada model serta proses pelatihan model yang dilakukan pada platform Google Colab.

Gambar 5. Skenario pengembangan model

Arsitektur yang dipilih dalam penelitian kali ini adalah arsitektur *Single Shot Detector (SSD)* dengan *MobileNet V2* sebagai *backbone*. Pemilihan arsitektur ini didasarkan pada kebutuhan akan model yang ringan dan efisien namun tetap memiliki performa yang baik dalam deteksi objek.

Pengembangan model dilakukan menggunakan *TensorFlow Object Detection API*. Langkah ini melibatkan pengkodean dan konfigurasi model sesuai dengan arsitektur yang telah dipilih. Peneliti menyesuaikan konfigurasi model seperti total langkah pelatihan, learning rate, optimizer dan hyperparameters lainnya untuk mengoptimalkan performa model.

Setelah tahap konfigurasi sudah dilakukan, proses selanjutnya adalah pelatihan model. Proses pelatihan model dilakukan di *Google Colab* menggunakan *GPU* untuk mempercepat komputasi. Pelatihan berlangsung hingga model mencapai konvergensi atau performa tidak meningkat signifikan. Model terbaik dipilih berdasarkan performa tertinggi pada set validasi untuk digunakan pada tahap pengujian dan implementasi.

E. Pengujian dan evaluasi model

Pada tahap ini, model yang telah dilatih akan diuji dan dievaluasi untuk mengukur kinerjanya dalam mendeteksi kerusakan jalan. Model diuji dengan memvariasikan beberapa parameter pelatihan untuk menemukan konfigurasi optimal:

a. Learning rate mengontrol seberapa besar model menyesuaikan diri saat proses pembelajaran - nilai yang terlalu besar membuat model belajar terlalu cepat dan tidak akurat, sedangkan nilai terlalu kecil membuat model belajar terlalu lambat [6].
b. Batch size adalah jumlah data yang diproses dalam satu iterasi pembelajaran, mempengaruhi kecepatan training dan penggunaan memori komputer [7].
c. Total step adalah jumlah iterasi pembelajaran yang dilakukan model selama proses training, menentukan seberapa lama model akan belajar dari data yang diberikan.
d. Split dataset adalah teknik pembagian data menjadi beberapa bagian (training, validation, testing) untuk memastikan model dapat dilatih dan dievaluasi secara terpisah [8].

Performa model kemudian dievaluasi menggunakan metrik evaluasi yang umum digunakan yaitu *precision, recall dan F1-score*. *Precision* mengukur seberapa akurat model dalam memprediksi kelas positif. Dengan kata lain, dari semua prediksi yang diberikan oleh model sebagai positif, berapa banyak yang benar-benar positif [9]. *Recall* mengukur seberapa baik model dalam menangkap semua sampel positif yang ada. Dengan kata lain, dari semua sampel yang sebenarnya positif, berapa banyak yang berhasil diprediksi dengan benar oleh model [10]. *F1-score* digunakan untuk mengukur keseimbangan antara *precision* dan *recall* dalam model klasifikasi atau deteksi. [11]

Metrik-metrik ini digunakan untuk menilai performa model dalam mendeteksi dan mengklasifikasikan kerusakan jalan dengan akurat dan efektif.

F. Konversi model

Model yang menunjukkan hasil dan performa terbaik akan dipilih dan dikonversi menjadi model dalam format Javascript menggunakan *Tensoflow JS converter,* agar dapat dijalankan dalam platform web.

III. Hasil dan Pembahasan

Deteksi kerusakan jalan dilakukan dengan memasukkan input berupa gambar melalui file gambar pada sistem menghasilkan klasifikasi kerusakan jalan sesuai dengan jenis kerusakan yang terdeteksi. Proses transfer learning dan fine tuning dilakukan untuk mendapatkan hasil yang optimal dan stabil untuk digunakan.

A. Hasil pengujian model

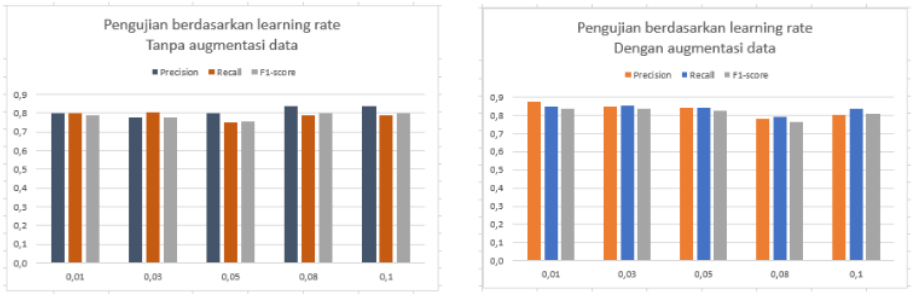
Hasil pengujian akan ditampilkan dalam bentuk tabel dan diagram untuk mempermudah proses analisis. Pengujian dilakukan dengan melakukan 5 kali percobaan untuk setiap konfigurasi parameter dan mengambil nilai rata-rata dari metrik precision, recall, dan F1-score sebagai indikator performa model. Tabel 1 akan menunjukkan hasil pengujian berdasarkan learning rate yang berbeda untuk menentukan konfigurasi optimal.

Tabel 1. Tabel pengujian berdasarkan *learning rate*

Jumlah Dataset	*Data Split (Train, valid, test)*	Total step	Augmentasi Data	*Batch Size*	*Learning rate*	*Precision*	*Recall*	*F1-score*	*Total loss*
1018	60, 25, 15	1000	Tidak	32	0,01	0,797	0,801	0,790	1,02
1600	85, 10, 5	1000	Ya	32	0,01	0,874	0,848	0,838	1,07
1018	60, 25, 15	1000	Tidak	32	0,03	0,777	0,807	0,777	0,64

1600	85, 10, 5	1000	Ya	32	0,03	0,849	0,850	0,835	0,52
1018	60, 25, 15	1000	Tidak	32	0,05	0,798	0,753	0,757	0,53
1600	85, 10, 5	1000	Ya	32	0,05	0,843	0,840	0,826	0,66
1018	60, 25, 15	1000	Tidak	32	0,08	0,840	0,788	0,797	0,68
1600	85, 10, 5	1000	Ya	32	0,08	0,778	0,791	0,762	0,88
1018	60, 25, 15	1000	Tidak	32	0,1	0,838	0,872	0,806	0,33
1600	85, 10, 5	1000	Ya	32	0,1	0,801	0,836	0,809	0,75

Hasil pengujian berdasarkan learning rate menunjukkan performa terbaik pada nilai 0,1 untuk model tanpa augmentasi data dengan precision 83,8%, recall 87,2% dan F1-score 80,6%. Hal ini menunjukkan bahwa model dapat melakukan update parameter dengan langkah yang lebih besar tanpa kehilangan kemampuan generalisasi. Namun, pada model dengan augmentasi data, justru learning rate yang lebih kecil (0,01) memberikan hasil optimal dengan precision 87,4%, recall 84,8%, dan F1-score 83,8%. Ini terjadi karena dengan dataset yang lebih besar dan beragam hasil augmentasi, model membutuhkan langkah update yang lebih kecil untuk mencapai konvergensi yang stabil. Perbedaan performa model akan ditampilkan pada diagram pada gambar 7.



Gambar 6. Diagram pengujian learning rate

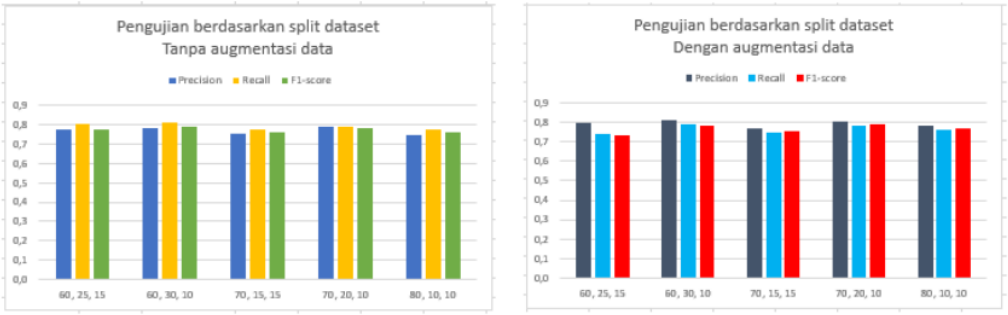
Setelah menganalisis hasil uji berdasarkan variasi learning rate, pembahasan berikutnya akan difokuskan pada hasil pengujian yang diperoleh dengan menggunakan variasi pembagian dataset. Data hasil pengujian tersebut disajikan dalam tabel 2 untuk dianalisis lebih lanjut.

Tabel 2. Tabel pengujian berdasarkan *split data*

Jumlah Dataset	***Data Split (Train, valid, test)***	**Total step**	**Augmentasi Data**	***Batch Size***	***Learning rate***	***Precision***	***Recall***	***F1-score***	***Total loss***
1018	60, 25, 15	1000	Tidak	32	0,03	0,777	0,807	0,776	0,51
1600	60, 25, 15	1000	Ya	32	0,03	0,797	0,742	0,734	0,54
1018	60, 30, 10	1000	Tidak	32	0,03	0,781	0,810	0,795	0,39
1600	60, 30, 10	1000	Ya	32	0,03	0,810	0,790	0,786	0,64
1018	70, 15, 15	1000	Tidak	32	0,03	0,752	0,780	0,765	0,63

1600	70, 15, 15	1000	Ya	32	0,03	0,768	0,748	0,755	0,65
1018	70, 20, 10	1000	Tidak	32	0,03	0,790	0,795	0,787	0,49
1600	70, 20, 10	1000	Ya	32	0,03	0,805	0,781	0,788	0,64
1018	80, 10, 10	1000	Tidak	32	0,03	0,745	0,775	0,760	0,66
1600	80, 10, 10	1000	Ya	32	0,03	0,785	0,765	0,772	0,53

Berdasarkan pengujian pembagian dataset, hasil terbaik untuk model tanpa augmentasi data dicapai pada pembagian 70:20:10 dengan precision 79%, recall 79,5% dan F1-score 78,7%. Pada model dengan augmentasi data (1600 data), performa terbaik diperoleh pada pembagian 60:30:10 dengan precision 81%, recall 79% dan F1-score 78,6%. Secara umum, model dengan augmentasi data menunjukkan performa yang lebih konsisten dan sedikit lebih baik dibandingkan tanpa augmentasi di berbagai variasi pembagian dataset, ini terlihat pada gambar 8.



Gambar 7. Diagram pengujian split dataset

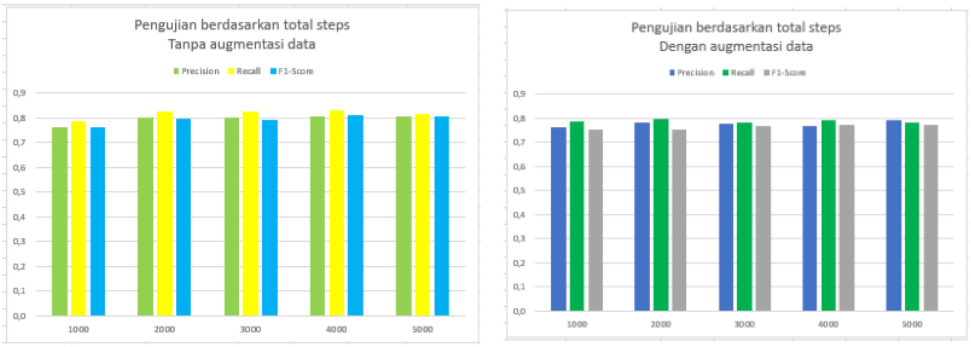
Berikutnya, pembahasan beralih pada analisis hasil uji yang melibatkan variasi total steps. Data hasil pengujian ini disajikan dalam tabel 3 untuk memberikan gambaran yang lebih jelas terkait pengaruh total steps terhadap performa model.

Tabel 3. Tabel pengujian berdasarkan *total steps*

Jumlah Dataset	***Data Split (Train, valid, test)***	***Total Steps***	**Augmentasi Data**	***Batch Size***	***Learning Rate***	***Precision***	***Recall***	***F1-Score***	***Total loss***
1018	60, 25, 15	1000	Tidak	32	0,01	0,765	0,789	0,762	0,59
1600	60, 25, 15	1000	Ya	32	0,01	0,763	0,786	0,753	0,48
1018	60, 25, 15	2000	Tidak	32	0,01	0,801	0,826	0,797	0,38
1600	60, 25, 15	2000	Ya	32	0,01	0,781	0,799	0,754	0,55
1018	60, 25, 15	3000	Tidak	32	0,01	0,800	0,828	0,794	0,51
1600	60, 25, 15	3000	Ya	32	0,01	0,776	0,781	0,767	0,38
1018	60, 25, 15	4000	Tidak	32	0,01	0,805	0,830	0,810	0,39

1600	60, 25, 15	4000	Ya	32	0,01	0,768	0,790	0,773	0,34
1018	60, 25, 15	5000	Tidak	32	0,01	0,808	0,819	0,806	0,32
1600	60, 25, 15	5000	Ya	32	0,01	0,790	0,785	0,775	0,32

Dalam pengujian berdasarkan total langkah pelatihan, model dengan augmentasi menunjukkan peningkatan performa hingga 5.000 langkah, dengan precision 79%, recall 78% dan F1-score 77%. Model tanpa augmentasi mencapai performa terbaik pada 5.000 langkah, dengan precision 80%, recall 81%, dan F1-score 80%. Kedua model memiliki total loss terendah sebesar 0,32 pada langkah tersebut.



Gambar 8. Diagram pengujian *total steps*

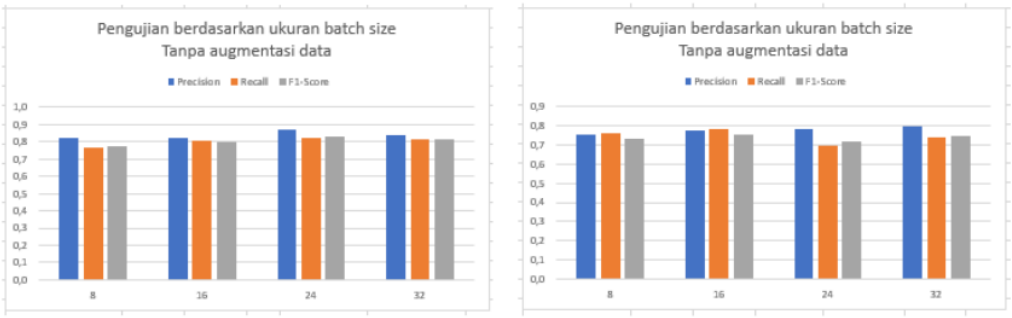
Tabel 4 akan menunjukkan hasil pengujian berdasarkan ukuran batch size yang digunakan untuk mengetahui perbedaan yang diberikan.

Tabel 4. Tabel pengujian berdasarkan ukuran *batch size*

Jumlah Dataset	***Data Split (Train, valid, test)***	**Total Step**	**Augmentasi Data**	***Batch Size***	***Learning Rate***	***Precision***	***Recall***	***F1-Score***	***Total loss***
1018	60, 25, 15	1000	Tidak	8	0,03	0,823	0,765	0,778	0,74
1600	60, 25, 15	1000	Ya	8	0,03	0,753	0,759	0,730	0,5
1018	60, 25, 15	1000	Tidak	16	0,03	0,826	0,808	0,800	0,46
1600	60, 25, 15	1000	Ya	16	0,03	0,778	0,784	0,755	0.57
1018	60, 25, 15	1000	Tidak	24	0,03	0,868	0,824	0,831	0,47
1600	60, 25, 15	1000	Ya	24	0,03	0,787	0,697	0,722	0,69
1018	60, 25, 15	1000	Tidak	32	0,03	0,836	0,814	0,815	0,49
1600	60, 25, 15	1000	Ya	32	0,03	0,797	0,742	0,744	0,57

Dalam pengujian berdasarkan batch size, model dengan augmentasi data mencapai performa optimal pada batch size 16 dengan precision 77%, recall 78% dan F1-score 75%. Sementara itu, model tanpa augmentasi

menunjukkan hasil terbaik pada batch size 24 dengan precision 86%, recall 82% dan F1-score 83%. Performa model cenderung menurun pada ukuran batch yang lebih besar, sebagaimana ditampilkan pada Gambar 10.

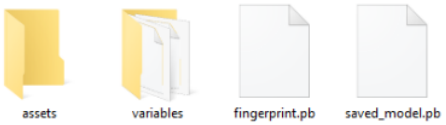


Gambar 9. Diagram garis pengujian ukuran *batch size*

Berdasarkan keseluruhan pengujian yang dilakukan, performa terbaik dicapai oleh model dengan augmentasi data menggunakan learning rate 0,01, yang menghasilkan precision 87,4%, recall 84,8%, dan F1-score 87,8%. Hasil ini menunjukkan bahwa kombinasi augmentasi data dan pemilihan learning rate yang tepat dapat meningkatkan kemampuan model dalam mendeteksi kerusakan jalan secara signifikan.

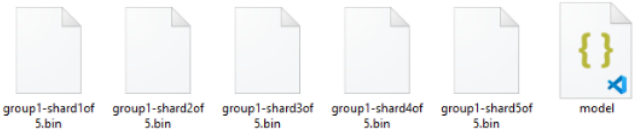
B. Konversi model

Setelah dilakukan evaluasi, model yang menunjukkan performa yang stabil antara precision, recall, dan loss pada data uji, dipilih untuk digunakan lebih lanjut. Model ini kemudian dikonversi ke format TensorFlow.js agar dapat diimplementasikan pada aplikasi berbasis web, memungkinkan deteksi kerusakan jalan dilakukan secara langsung melalui browser.



Gambar 10. Model hasil pelatihan

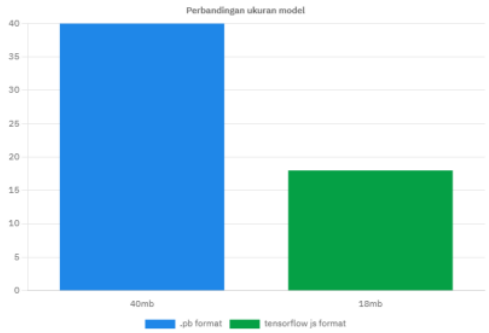
Gambar 9 memperlihatkan format model *TensorFlow* yang disimpan dengan ekstensi .pb (*Protocol Buffer*), yang merupakan format standar untuk menyimpan model *deep learning*. Pemilihan format ini sejalan dengan penelitian [12] yang juga memanfaatkan platform TensorFlow untuk implementasi model deteksi kerusakan jalan, dimana format ini terbukti efektif dalam menyimpan graph model dan parameter-parameternya secara efisien.



Gambar 11. Model hasil konversi

Sementara itu, Gambar 10 menunjukkan hasil konversi model ke format TensorFlow.js yang memungkinkan eksekusi model langsung di browser. Untuk mengintegrasikan model ke dalam web, diperlukan dua file utama hasil konversi yaitu model.json yang berisi arsitektur model dan weight.bin yang menyimpan bobot model. File model.json berperan sebagai manifest yang mendeskripsikan struktur model dan referensi ke file weight.bin. Kedua file ini kemudian dimuat menggunakan API TensorFlow.js yang memungkinkan model dapat dijalankan langsung di browser menggunakan WebGL. Pendekatan konversi ke TensorFlow.js ini merupakan pengembangan dari penelitian sebelumnya [13], dimana model masih menggunakan implementasi berbasis

server. Dengan implementasi berbasis browser, sistem dapat melakukan komputasi di sisi client yang berpotensi meningkatkan efisiensi pemantauan kondisi jalan sekaligus mengurangi kebutuhan komputasi server dan meminimalkan latency dalam proses deteksi kerusakan jalan.

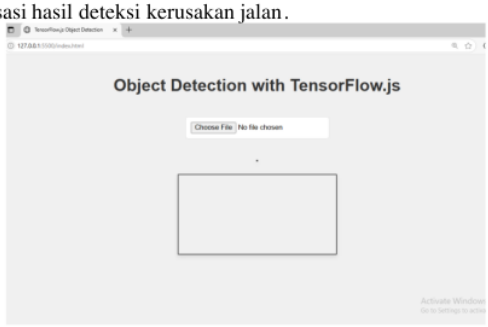


Gambar 12. Perbandingan ukuran model

Proses konversi menghasilkan perubahan ukuran model yang cukup signifikan, dimana model awal dalam format .pb memiliki ukuran 40 MB, setelah dikonversi menjadi format TensorFlow.js ukurannya menjadi 18 MB. Perubahan ukuran ini terjadi karena proses optimasi dan kuantisasi yang dilakukan oleh TensorFlow.js converter, namun tetap mempertahankan kemampuan model dalam melakukan deteksi.

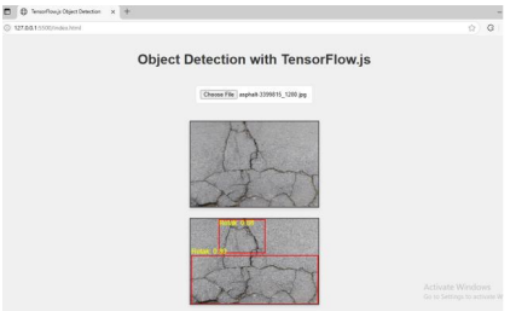
C. Tampilan web

Gambar 11 menunjukkan tampilan antarmuka web yang telah dikembangkan, dengan fitur utama berupa area unggah gambar dan visualisasi hasil deteksi kerusakan jalan.



Gambar 13. Tampilan input gambar

Gambar 12 menunjukkan hasil deteksi ketika gambar telah berhasil dimasukkan oleh pengguna. Sistem akan memuat model untuk mendeteksi jenis kerusakan jalan dan menampilkan hasilnya dalam bentuk bounding box pada area kerusakan beserta confidence score yang menunjukkan tingkat keyakinan deteksi..



Gambar 14. Tampilan hasil deteksi gambar

IV. Simpulan

Penelitian ini bertujuan untuk mengembangkan model deteksi kerusakan jalan menggunakan arsitektur SSD MobileNet yang dapat mengidentifikasi jenis kerusakan dengan presisi dan keakuratan tinggi. Hasil penelitian menunjukkan bahwa model yang diimplementasikan berhasil mencapai performa terbaik pada model dengan augmentasi data dan konfigurasi learning rate 0,01, dengan hasil *precision* 87,4%, *recall* 84,8%, dan *F1-score* 83,8%.

Model ini juga berhasil dikonversi ke format TensorFlow.js, memungkinkan penerapan pada aplikasi berbasis web yang praktis dan mudah diakses. Dengan konfigurasi yang optimal, model mampu memberikan deteksi yang akurat pada kondisi pengambilan gambar tertentu, meskipun performa menurun pada gambar dengan visibilitas rendah atau sudut pengambilan yang tidak ideal.

Untuk pengembangan selanjutnya, penelitian ini dapat ditingkatkan dengan menambahkan variasi data pelatihan untuk menangani berbagai kondisi gambar yang lebih kompleks. Selain itu, pengembangan kemampuan model untuk mengklasifikasikan jenis dan tingkat keparahan kerusakan jalan secara lebih rinci dapat menjadi langkah lanjutan untuk meningkatkan manfaat praktis model ini.

.

Ucapan Terima Kasih

Terimakasih kepada Alvaro Basily yang telah menyediakan dataset gambar kerusakan jalan yang digunakan dalam penelitian ini.

Referensi

[1] I. Himmati, I.P.Prawesti,and A.A.Brahmantya, "roadect:deteksi persoalan infrastruktur jalan sebagai solusi strategi igital dalam penyampaian aspirasi masyarakat berkelanjutan" 2023.

[2] A. N. Utomo and N. Lestari, "APLIKASI DETEKSI KERUSAKAN JALAN RAYA MENGGUNAKAN ALGORITMA K-NN (K-NEAREST NEIGHBOUR) ROAD DETECTION APPLICATION USING K-NN LGORITHM (K-NEAREST NEIGHBOUR)," vol. 10, no. 1, 2021.

[3] P. R. Aningtiyas, A. Sumin, and S. Wirawan, "Pembuatan Aplikasi Deteksi Objek Menggunakan TensorFlow Object Detection API dengan Memanfaatkan SSD MobileNet V2 Sebagai Model Pra - Terlatih," *Jurnal Ilmiah omputasi*, vol. 19, no. 3, Mar. 2020, doi: 10.32409/jikstik.19.3.68.

[4] P. D. Arnesia, N. A. Pratama, and F. Sjafrina, "APLIKASI ARTIFICIAL INTELLIGENCE UNTUK MENDETEKSI OBJEK BERBASIS WEB MENGGUNAKAN LIBRARY TENSORFLOW JS, REACT JS DAN COCO DATASET," 2022.

[5] Basily, "Road Damage," https://www.kaggle.com/datasets/alvarobasily/road-damage.

[6] K. Kualitas Buah Salak dengan Transfer Learning Arsitektur VGG and A. Luthfiarta, "VGG16 Transfer Learning Architecture for Salak Fruit Quality Classification," *Jurnal Informatika dan Teknologi Informasi*, vol. 18, no. 1, pp. 37–8, 2021, doi: 10.31515/telematika.v18i1.4025.

[7] A. Julianto, A. Sunyoto, D. Ferry, and W. Wibowo, "OPTIMASI HYPERPARAMETER CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI PENYAKIT TANAMAN PADI (OPTIMIZATION OF CONVOLUTIONAL NEURAL NETWORK HYPERPARAMETERS FOR CLASSIFICATION OF RICE PLANT SEASES)." [Online]. Available: https://www.kaggle.com/tedisetiady/leaf-rice-dis-

[8] E. Muningsih, "KOMBINASI METODE K-MEANS DAN DECISION TREE DENGAN PERBANDINGAN RITERIA DAN SPLIT DATA," 2022.

[9] Z. Syahputra, "Penerapan SSD-Mobilenet Dalam Identifikasi Jenis Buah Apel," *Indonesian Journal of Education d Computer Science*, vol. 1, no. 1, 2023.

[10] C. Anwar, "Deteksi Objek Berbasis Web Menggunakan Tensorflow Js dan Coco Dataset pada Framework React J 2022.

[11] "Klasifikasi P derita Diabetes menggunakan Algoritma Machine Learning dan Z-Score".

[12] A. Pratama, "Deteksi Kerusakan Jalan Menggunakan Convolutional Neural Network Berbasis SSD EfficientNet rtikel Ilmiah," 2020.

[13] B. Sasmito, B. H. Setiadji, and R. Isnanto, "Deteksi Kerusakan Jalan Menggunakan Pengolahan Citra Deep Learning di Kota Semarang," *TEKNIK*, vol. 44, no. 1, pp. 7–14, May 2023, doi: 10.14710/teknik.v44i1.51908.

New Microsoft Word Document.pdf

ORIGINALITY REPORT

12%	11%	8%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ejournal.itn.ac.id Internet Source	1%
2	Nabila Khairunisa, Carudin ., Asep Jamaludin. "ANALISIS PERBANDINGAN ALGORITMA CNN DAN YOLO DALAM MENGIDENTIFIKASI KERUSAKAN JALAN", Jurnal Informatika dan Teknik Elektro Terapan, 2024 Publication	1%
3	eprints.uad.ac.id Internet Source	1%
4	www.jatit.org Internet Source	1%
5	kc.umn.ac.id Internet Source	1%
6	jurnal.peneliti.net Internet Source	1%
7	Dadang Iskandar Mulyana, Ilham Wahyudi. "Deteksi Kerusakan Jalan Berdasarkan Citra Digital Menggunakan Convolutional Neural Network (CNN)", Jurnal Indonesia :	1%

	Manajemen Informatika dan Komunikasi, 2025 Publication	
8	jsisfotek.org Internet Source	1 %
9	ejurnal.seminar-id.com Internet Source	<1 %
10	ejurnal.umri.ac.id Internet Source	<1 %
11	elibrary.nusamandiri.ac.id Internet Source	<1 %
12	www.researchgate.net Internet Source	<1 %
13	www.scilit.net Internet Source	<1 %
14	repository.uksw.edu Internet Source	<1 %
15	ejournal.pelitaindonesia.ac.id Internet Source	<1 %
16	Pravendra Singh, Pratik Mazumder, Mohammed Asad Karim, Vinay P. Namboodiri. "Calibrating feature maps for deep CNNs", Neurocomputing, 2021 Publication	<1 %
17	repo.unand.ac.id Internet Source	

		<1%
18	ar.scribd.com Internet Source	<1%
19	datasetninja.com Internet Source	<1%
20	ejournal.jak-stik.ac.id Internet Source	<1%
21	ejournal.nusamandiri.ac.id Internet Source	<1%
22	es.scribd.com Internet Source	<1%
23	text-id.123dok.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		