TRANSMISI: JURNAL ILMIAH TEKNIK ELEKTRO, 28, (3), JULI 2026
p-ISSN 1411-0814 e-ISSN 2407-6422

PURWARUPA SISTEM KENDALI KECEPATAN KERETA BERBASIS KLASIFIKASI INSPEKSI REL MENGGUNAKAN FASTER R-CNN

Yurixa Sakhinatul Putri[1*)], Yul Yunazwin Nazaruddin[2)] dan Endang Juliastuti[2)]

[1]Program Studi Instrumentasi dan Kontrol Industri, Politeknik Negeri Jakarta, Depok, Indonesia
[2]Departemen Instrumentasi dan Kontrol, Institut Teknologi Bandung, Bandung, Indonesia

[*]*Penulis korespondensi, E-mail: yurixa.sakhinatul.putri@elektro.pnj.ac.id*

Abstrak

Inspeksi visual manual pada rel kereta api rentan terhadap kelelahan operator dan subjektivitas, yang berpotensi menyebabkan kesalahan deteksi. Penelitian ini mengembangkan purwarupa sistem kendali kecepatan kereta berbasis klasifikasi inspeksi rel menggunakan algoritma *Faster R-CNN*. Sistem terdiri dari kamera sebagai sensor visual, komputer sebagai *client* untuk melakukan proses klasifikasi objek, dan mikrokontroler sebagai *server* untuk mengendalikan kecepatan pada motor kereta. *Dataset* terdiri dari tiga kelas yaitu *broken* (rel putus), *curved* (rel melengkung), dan *erosion* (penghalang pada rel) dengan *dataset* terbatas 370 gambar. Proses pelatihan menggunakan *GPU Nvidia GeForce MX*130, *RAM* 6 *GB*, dengan total 7063 *step* dan waktu ± 3 jam. Didapatkan nilai akurasi 80%, *precision* 85%, *recall* 93%, dan *F1 score* 88%. Hasil klasifikasi diintegrasikan ke dalam sistem kendali kecepatan motor kereta melalui komunikasi TCP/IP. Sistem berhasil mendeteksi objek secara *real-time* dan mengontrol kecepatan motor kereta dengan logika *if else* sesuai tingkat kepercayaan hasil klasifikasi. Pengujian juga menunjukkan bahwa jarak deteksi objek dipengaruhi oleh posisi kamera terhadap jalur rel. Purwarupa ini dapat menunjukkan potensi yang baik dalam pengintegrasian sistem visi komputer dan kendali motor kereta api sebagai solusi untuk meningkatkan keselamatan pada kereta api.

Kata kunci: inspeksi rel, *faster r-cnn*, *kendali kecepatan*, sistem visi komputer, *TCP/IP*

Abstract

Manual visual inspection of railway tracks frequently encounters challenges such as operator fatigue and subjective assessments, which may lead to detection errors. This study proposes a prototype of a train speed control system based on rail inspection classification utilizing the Faster R-CNN algorithm. The system comprises a camera serving as a visual sensor, a computer functioning as a client to perform object classification, and a microcontroller acting as a server to control the train motor speed. The dataset consists of three classes: ***broken*** *(fractured rail),* ***curved*** *(bent rail), and* ***erosion*** *(obstruction on the track), with a total of 300 images. The training process was conducted using an Nvidia GeForce MX130 GPU and 6 GB RAM, completing 7,063 steps within approximately 3 hours. The model achieved an accuracy of 80%, precision of 85%, recall of 93%, and an F1-score of 88%. The classification results were integrated into the train motor speed control system via TCP/IP communication. The system successfully performed real-time object detection and controlled the train motor speed using if-else logic based on the classification confidence score. Experimental results also indicate that the object detection range is influenced by the camera's positioning relative to the railway track. This prototype demonstrates strong potential for the integration of computer vision and train motor control systems as a means to enhance railway safety.*

Keywords: railway inspection, Faster R-CNN, speed control, computer vision, TCP/IP

1. Pendahuluan

Kereta api adalah salah satu moda transportasi massal yang banyak diminati masyarakat karena memberikan banyak keuntungan. Selain dapat mengangkut penumpang dalam jumlah besar, kereta api juga relatif cepat, nyaman, dan hemat biaya operasional. Hal ini menjadikan pilihan yang popular, terlebih ketika mobilitas harian dan perjalanan jauh semakin meningkat seperti di Indonesia [1].

Dengan meningkatnya pengguna kereta api, keselamatan operasional menjadi tantangan yang serius. Salah satu faktor keselamatan yang harus diperhatikan ialah pemeliharaan rel kereta api. Karena, infrastruktur rel yang tidak dipelihara dapat mengalami kerusakan [2] seperti patah atau bergelombang yang berpotensi membahayakan operasi layanan [12]. Studi di Divre IV Tanjung Karang juga mengungkapkan bahwa rel patah akibat dari kondisi lingkungan ekstrim seperti hujan lebat dan ekspansi tanah.

Kejadian ini tentu akan meningkatkan risiko terjadinya kecelakaan dan menjadi hal yang perlu diantisipasi [3].

Untuk menjaga keamanan perjalanan, petugas inspeksi rel melakukan pemeriksaan secara berkala terhadap kondisi rel. Inspeksi visual manual dilakukan dengan menyusuri rel atau menggunakan kereta inspeksi untuk mendeteksi retakan, keausan, benda asing, dan kondisi fastener. Namun, metode ini menuntut beban fisik tinggi, bergantung pada konsentrasi manusia, dan rentan terhadap kelelahan operator. Sekitar 67% kesalahan inspeksi disebabkan oleh gangguan perhatian seperti kelelahan dan *inattentional blindness* [4]. Selain itu, inspeksi visual bersifat subjektif karena penilaian tingkat bahaya kerusakan sangat bergantung pada pengalaman dan kondisi pengamat. [5].

Sebagai upaya peningkatan kualitas inspeksi visual, kamera digital kini telah dipasang pada armada kereta atau kereta inspeksi. Kamera ini merekam kondisi rel secara terus-menerus selama perjalanan. Meskipun perekaman berjalan baik, analisis citra rel masih banyak dilakukan secara manual atau dengan perangkat lunak yang belum otomatis [5]. Akibatnya, deteksi kerusakan rel belum optimal dan tidak dapat dilakukan secara *real-time*. Faktanya, kecepatan respons deteksi ialah sangat krusial dalam menjamin keselamatan operasional. Oleh karena itu, solusi inspeksi cerdas perlu diterapkan seperti penerapan sistem visi komputer dalam menginspeksi rel.

Salah satu metode yang digunakan pada sistem visi komputer adalah jaringan saraf tiruan (JST) atau *neural network* (NN). Metode ini memiliki kemampuan untuk menghasilkan keputusan atau prediksi berdasarkan proses pelatihan berulang terhadap kumpulan data (seperti citra) sehingga model dapat mengenali pola atau objek tertentu secara akurat [6]. Jaringan saraf tiruan konvensional masih bergantung pada ekstraksi fitur manual melalui tahap pra-pemrosesan [12]. Hal ini disebabkan oleh keterbatasan model dalam menangkap fitur penting secara langsung dari data asli. Oleh karena itu, dibutuhkan tahap pra-pemrosesan (*pre-processing*) untuk mengekstraksi fitur-fitur citra seperti bentuk, warna, dan tekstur yang kemudian digunakan sebagai input pada JST sehingga mesin dapat dengan mudah melakukan pembelajaran [6].

Namun, dengan kemajuan teknologi komputasi, JST berlapis banyak (*deep neural network*) mampu melakukan ekstraksi fitur secara otomatis dari data mentah [6]. Proses pembelajaran yang dilakukan oleh JST berlapis banyak inilah yang dinamakan *deep learning* [13].

Salah satu metode *deep learning* untuk deteksi objek dan klasifikasi citra ialah metode jaringan syaraf konvolusional (JSK) atau *convolutional neural network* (*CNN*). *CNN* mampu mengekstraksi fitur visual dan pola spasial secara otomatis melalui lapisan konvolusi yang kompleks [7]. Dalam perkembangannya, arsitektur CNN terus disempurnakan untuk meningkatkan kecepatan dan akurasi deteksi [14]. Salah satu model yang paling banyak digunakan dalam bidang deteksi objek adalah *Faster R-CNN* yang memanfaatkan *Region Proposal Network* (RPN) untuk mengusulkan wilayah objek secara otomatis [15], sehingga deteksi dapat dilakukan secara *end-to-end* dan *real-time*.

Beberapa penelitian sebelumnya menunjukkan keberhasilan *Faster R-CNN* dalam mendeteksi objek pada berbagai domain, termasuk dalam bidang transportasi. Seperti pada studi oleh Jasman (2022) yang telah mendeteksi objek kereta api untuk mencegah terjadinya kecelakaan di persimpangan jalan raya menggunakan *Faster R-CNN*, memperoleh rata-rata akurasi diatas 90% [8]. Studi oleh Enny (2023) mengembangkan system inspeksi otomatis untuk mendeteksi cacat pada bentuk roda gigi (*gear*) menggunakan *Faster R-CNN* dengan akurasi rata-rata 83,32% [9]. Selain itu, studi oleh Merve (2024) juga mengembangkan sistem inspeksi rel menggunakan *Faster R-CNN* untuk mengklasifikasikan cacat permukaan rel dan *fastener* dengan hasil akurasi mencapai 98% untuk deteksi cacat permukaan rel dan 95% untuk deteksi cacat pada *fastener* [10].

Namun, sebagian besar penelitian tersebut hanya terbatas pada tahap klasifikasi atau deteksi objek [11], [17], [18], tanpa adanya integrasi terhadap sistem kontrol pada operasional kereta. Padahal, informasi hasil klasifikasi perlu ditindaklanjuti secara cepat untuk mencegah risiko kecelakaan. Oleh karena itu, dibutuhkan sistem yang tidak hanya mampu melakukan deteksi dan klasifikasi kerusakan rel, tetapi juga mampu mengendalikan kecepatan kereta berdasarkan hasil klasifikasi tersebut secara *real-time*.

Kebaruan (*novelty)* penelitian ini terletak pada pengembangan purwarupa yang mengintegrasikan *Faster R-CNN* dengan sistem kendali kecepatan kereta secara real-time. Citra rel yang diperoleh dari kamera diproses untuk mendeteksi dan mengklasifikasikan kerusakan rel serta objek penghalang. Hasil klasifikasi berupa nilai tingkat kepercayaan digunakan secara langsung sebagai dasar pengaturan sinyal PWM pada motor penggerak. Sehingga, sistem mampu melakukan respons kecepatan kereta secara otomatis sesuai kondisi rel yang terdeteksi. Pembuatan purwarupa ini sangat penting sebagai langkah awal untuk menguji keandalan sistem dalam kondisi nyata dan menilai performa integrasi antara teknologi visi komputer dan sistem kontrol pada kereta. Dengan adanya purwarupa, pengujian fungsionalitas dan validasi awal terhadap konsep sistem dapat dilakukan sebelum diterapkan secara penuh dalam lingkungan operasional.

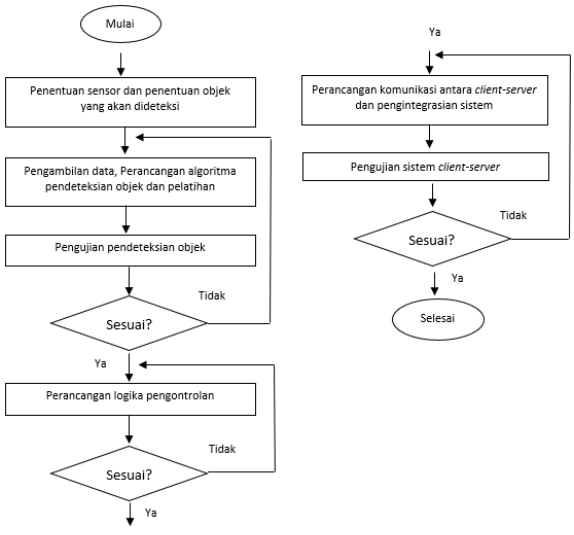
Penelitian ini bertujuan untuk mengembangkan dan menguji purwarupa sistem kendali kecepatan kereta berbasis klasifikasi inspeksi rel menggunakan *Faster R-CNN*, serta mengevaluasi performa sistem dalam

mendeteksi kondisi rel dan meresponsnya melalui pengendalian kecepatan kereta secara *real-time*. Kontribusi penelitian ini meliputi perancangan integrasi *end-to-end* antara sistem visi komputer berbasis *deep learning* dan sistem kendali kecepatan kereta, implementasi mekanisme pengambilan keputusan kendali berbasis tingkat kepercayaan hasil klasifikasi, serta penyajian hasil evaluasi kinerja sistem deteksi dan respons kendali pada purwarupa kereta api.

2. Metode

Perancangan sistem pendeteksi cacat dan penghalang rel berbasis *deep learning* menggunakan *faster r-cnn* dilakukan dengan alat dan bahan utama berupa set miniatur kereta api LEGO, *webcam Logitech* 30 *FPS*, pasir mainan, *driver motor*, mikrokontroler *NodeMCU*, batterai 12 volt dan laptop.

Rancangan sistem yang dibuat terdiri dari *hardware* dan *software*. *Hardware* berupa perangkat pendeteksi, perangkat pemroses, dan perangkat pengirim. Sedangkan untuk *software* berupa perancangan algoritma pengenalan objek, perancangan pengontrolan kecepatan motor kereta api dan pengintegrasian sistem dalam pengambilan pengontrolan. Pengintegrasian sistem dilakukan melalui protokol komunikasi TCP/IP. Dalam kedudukannya dikenal istilah *client* dan *server*.

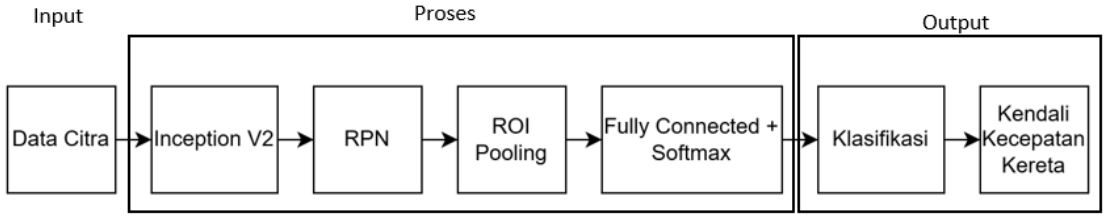


Gambar 1. Diagram Alir Penelitian

Client merupakan sebuah istilah pada komunikasi yang berperan dalam memberikan suatu perintah atau permintaan kepada *server* dalam melakukan sebuah tindakan berupa penurunan atau peningkatan kecepatan motor. Pada bagian *client* ini digunakan sebagai tempat untuk mendeteksi objek dengan sensor kamera digital. Sensor kamera disini bekerja pada daerah rentang spektrum *visible*. Saat adanya cahaya, cahaya akan masuk ke kamera melalui lensa kemudian cahaya yang ditangkap akan terfokuskan dan menghasilkan informasi digital berupa citra RGB. Citra RGB yang diinginkan berupa kerusakan dan penghalang pada rel kereta api. Informasi ini akan diolah melalui teknologi komputer visi menggunakan algoritma pendeteksi objek dengan proses pembelajaran berulang-ulang sehingga dapat mengenali 3 kelas objek berupa *broken*, *curved* dan *erosion*. Setelah sistem dapat mengenali atau mendeteksi citra, informasi yang didapat akan diubah melalui bilangan *integer* dan dikirimkan ke mikrokontroler melalui sebuah protokol komunikasi TCP/IP. Disinilah peran mikrokontroler sebagai *server*. Diagram alir pada penelitian ini dapat dilihat pada Gambar 1.

Pada bagian *server* digunakan sebagai tempat mikrokontroler untuk memberikan keputusan atau tindakan terhadap motor DC agar menaikkan atau menurunkan kecepatan motor pada kereta api berupa sinyal PWM. Informasi yang diperoleh akan menentukan besarnya sinyal PWM yang akan dibaca oleh mikrokontroler. Hasil pembacaan yang berupa sinyal *digital* akan diubah menjadi sinyal *analog* (*digital to analog converter*) dan akan menggerakkan kecepatan motor DC pada kereta api melalui modul penggerak motor.

Alur kerja sistem dapat dilihat pada Gambar 2. Proses dimulai dari data citra yang diperoleh sebagai masukan sistem. Citra tersebut diproses menggunakan *Inception V2* yang berfungsi sebagai *backbone network* untuk mengekstraksi fitur visual penting dari objek. Fitur yang dihasilkan selanjutnya diproses oleh Region Proposal Network (RPN) untuk menghasilkan kandidat wilayah objek yang berpotensi mengandung cacat atau penghalang. Setiap kandidat wilayah kemudian dinormalisasi ukurannya melalui tahap ROI *pooling* sebelum diklasifikasikan menggunakan lapisan *fully connected* dengan fungsi aktivasi *softmax*. Kemudian, hasil klasifikasi yang berupa kelas objek dan tingkat kepercayaan selanjutnya digunakan sebagai dasar pengambilan keputusan kendali kecepatan kereta.

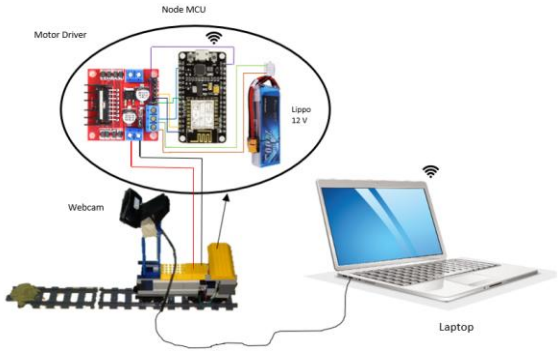


Gambar 2. Diagram Blok Arsitektur Sistem

2.1. Skema Perancangan Sistem

Skema perancangan sistem dapat dilihat pada Gambar 3. Objek yang akan dideteksi berupa 3 kelas objek yaitu *broken* (rel retak atau terputus), *curved* (rel melengkung dan *erosion* (pasir mainan yang diasumsikan adanya longsor sebagai penghalang didepan rel). Objek tersebut akan ditangkap melalui kamera kemudian diolah di komputer melalui *framework Tensorflow* dengan bahasa pemrograman *python*. Hasil pengolahan tersebut berupa nilai *score* yang merepresentasikan tingkat kepercayaan. Dari nilai *score* tersebut akan di berikan logika *if else* untuk memberikan perintah yang dikelompokkoan dalam 3 kondisi berupa gas, berhenti atau hati-hati. Kondisi-kondisi yang berupa nilai *integer* tersebut akan dikirimkan secara nirkabel dari komputer yang sebagai *client* ke mikrokontroler *NodeMCU* sebagai *server* melalui protokol komunikasi TCP/IP.

Didalam mikrokontroler yang diprogram melalui bahasa pemrograman *Arduino* akan menerima kiriman dari *client* dalam satu jaringan yang sama berupa nilai-nilai yang merepresentasikan kondisi-kondisi. Kondisi-kondisi tersebut memberikan nilai digital berupa PWM yang telah ditetapkan dimasing-masing kondisi. Semakin besar nilai PWM maka semakin besar nilai *duty cycle* dan semakin besar pula nilai kecepatan motor. Sinyal kontrol yang diberikan dari PWM tersebut kemudian akan diolah pada modul penggerak motor dan memberikan keluaran pada kecepatan motor untuk bergerak, berhenti maupun pelan.



Gambar 3. Skema Perancangan Sistem

2.2. Perancangan Algoritma Pendeteksi Objek

Perancangan algoritma untuk mendeteksi objek dilakukan dalam beberapa tahapan yaitu pengambilan data, *pre-processing* data, pelatihan data dan pengujian data. Model yang digunakan untuk mendeteksi objek menggunakan model arsitektur *faster R-CNN* dengan *backbone network* Inception V2.

2.2.1. Pengambilan Dataset

Dataset dipersiapkan sebagai *input* terhadap model dalam melakukan pembelajaran. Pada penelitian ini pengambilan gambar dataset menggunakan kamera telepon genggam. Hal ini dikarenakan resolusi kamera telepon genggam lebih baik daripada kamera *webcam*. Objek yang akan diklasifikasikan menjadi 3 kelas. Kelas pertama adalah rel retak atau putus yang dilabeli "*broken"*, kelas kedua adalah rel bengkok yang dilabeli "*curved*" dan kelas ketiga adalah adanya penghalang didepan rel berupa pasir mainan yang diasumsikan longsor dengan dilabeli "*erosion"*. Rel bengkok didisain khusus dan dicetak melalui mesin pencetak 3 dimensi. Jenis-jenis dataset dapat dilihat pada Gambar 4.

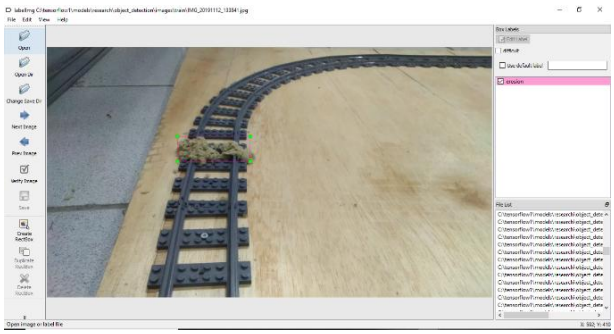
a b c

Gambar 4. Jenis Dataset (a) broken, (b) curved, dan (c) erosion

Pengambilan dataset berjumlah 370 gambar untuk data *training* dengan pembagian masing-masing kelas yang proporsional dan data *testing* berjumlah 70 gambar. Dalam pengambilan gambar masing-masing kelas diambil dengan berbagai variasi posisi, variasi lebar keregangan rel yang putus dan variasi ketinggian longsor.

2.2.2. *Pre-processing* Data

Agar sistem didalam proses pembelajaran tidak terlalu berat maka semua gambar yang semula berukuran 2688 x 1512 pixel dikecilkan menjadi 672 X 378 pixel.



Gambar 5. Proses Pemberian *Bounding Boxes* dan Pelabelan Dataset

Setelah itu dilakukan pelabelan data menggunakan *tools* LabelImg. Pelabelan dilakukan dengan memberikan *bounding boxes* atau kotak pembatas terhadap objek yang akan dideteksi. Hasil pemberian label ini akan digabungkan dalam sebuah file berkestensi *.xml* yang berupa nama, nilai dimensi, kelas dan posisi koordinat gambar. Kemudian file tersebut dikonversikan menjadi

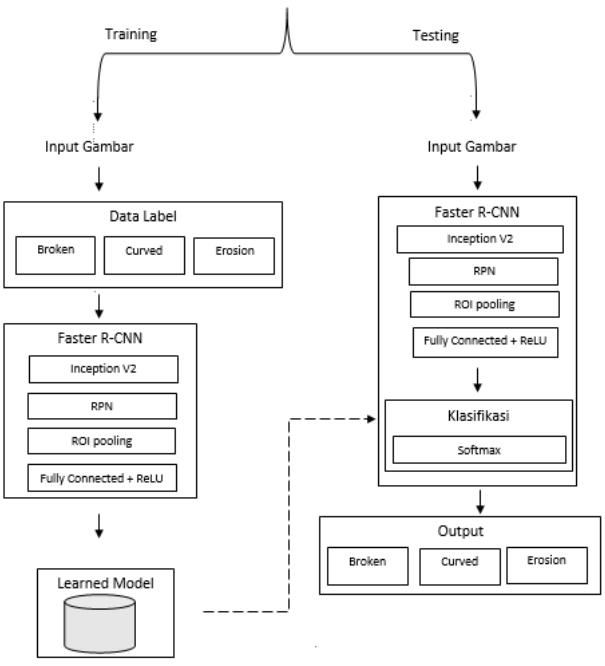
sebuah data bertipe *binary string* yang berekstensi *.csv* agar dapat dibaca dan diolah oleh komputer. Contoh proses pelabelan dataset dapat dilihat pada Gambar 5.

Setelah melakukan pelabelan, terakhir adalah mengkonfigurasikan peta label (*label map*) sesuai dengan banyaknya kelas yaitu 3 buah kelas dan disimpan dalam bentuk file *.pbtx*.

2.2.3. Pelatihan Data

Proses pelatihan data dilakukan melalui proses konfigurasi pada suatu file *pre-trained* model terlebih dahulu dengan model *faster R-CNN* dan *backbone network* Inception V2. Diagram alir proses pelatihan dapat ditunjukkan pada Gambar 6. Pada pelatihan ini digunakan ukuran *batch* sebesar satu, *learning rate* sebesar 0.002 dan augmentasi data.

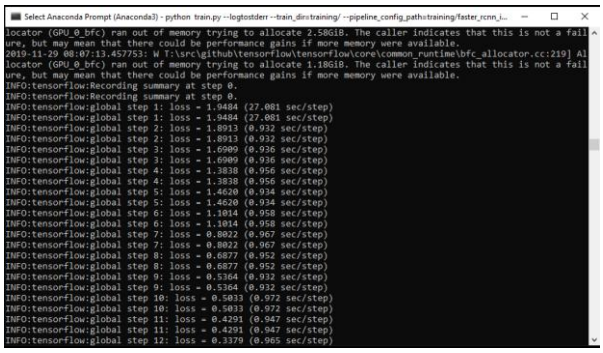
Ukuran *batch* menunjukkan jumlah data yang diproses dalam satu iterasi pelatihan. Dengan *batch size* bernilai satu, setiap iterasi pelatihan memproses satu citra dari total 300 citra data *training*. Pemilihan *batch size* bernilai satu ini dilakukan dengan mempertimbangkan keterbatasan sumber daya komputasi serta karakteristik arsitektur *Faster R-CNN*, di mana setiap citra menghasilkan sejumlah besar region proposal yang membutuhkan memori relatif besar. Penggunaan *batch size* bernilai satu memungkinkan proses pelatihan berjalan stabil dan membantu model mencapai konvergensi secara bertahap pada dataset berukuran terbatas. Serta agar memori yang diperlukan proses kecil dan akan lebih cepat konvergen.



Gambar 6. Diagram Blok Arsitektur Sistem

Learning rate atau laju pembelajaran merupakan nilai langkah yang digunakan dalam proses pembelajaran atau dapat dikatakan sebuah parameter pada *gradient descent* untuk mengoptimalkan iterasi dengan merepresentasikan nilai *loss function*. Dengan kata lain, *learning rate* digunakan sebagai parameter pada *gradient descent* untuk mengatur besar langkah pembaruan bobot model. Semakin kecil nilai *learning rate* artinya semakin pendek langkah yang dilakukan untuk pembelajaran maka semakin kecil pula (*cost function* atau *loss function*) kerugian yang akan didapat. Nilai *learning rate* yang relatif kecil ini dipilih untuk menjaga stabilitas proses pelatihan dan menghindari fluktuasi nilai *loss* yang berlebihan, khususnya pada proses *fine-tuning* model pra-latih. Kemudian pada pelatihan ini juga digunakan augmentasi data berupa pembalikkan gambar horizontal (*flip* horizontal). Hal ini dilakukan agar menambah variasi data pelatihan sehingga model menjadi lebih *robust* terhadap variasi orientasi objek dan mampu meningkatkan kemampuan generalisasi sistem deteksi.

Spesifikasi komputer yang digunakan pada proses pelatihan menjadi salah satu parameter yang akan menentukan besarnya akurasi. Dalam pelatihan ini digunakan laptop dengan spesifikasi *GPU Nvidia GeForce MX* 130 dan *RAM* 6 *GB*. Selama proses pelatihan ditampilkan jumlah langkah pelatihan, nilai *loss* dan waktu proses pelatihan per langkah. Tampilan proses pelatihan dapat diperlihatkan pada Gambar 7.



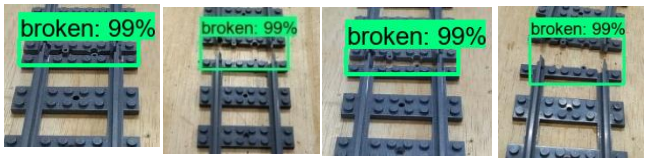
Gambar 7. Proses Awal Pelatihan

Dapat dilihat bahwa pada langkah awal pelatihan sistem memerlukan waktu yang lebih besar (27.081 *sec/step*) dibandingkan langkah selanjutnya (0.9 *sec/step*). Hal ini dikarenakan sistem perlu mengenali masukan gambar dan mendapatkan bobot awal pembelajaran telebih dahulu. Ketika sudah mendapat bobot maka proses melakukan *back propagation* atau propagasi balik sampai mendapat nilai *loss* sekecil mungkin menggunakan metode *gradient-descent*.

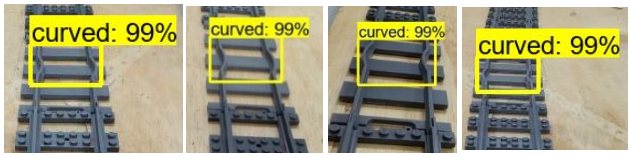
2.2.4. Pengujian Data

Setelah melakukan pelatihan data, langkah selanjutnya adalah melakukan pengujian data sebagai proses

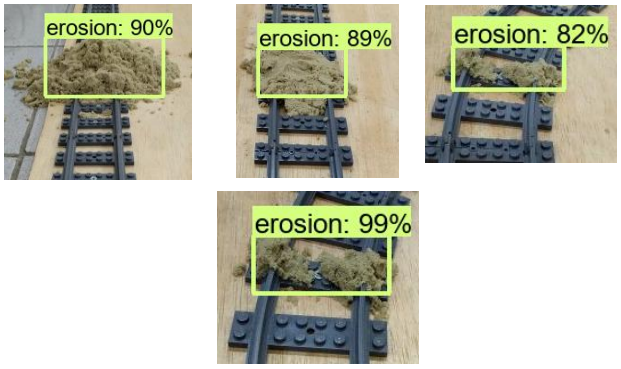
pengklasifikasian gambar. Pada tahap ini dapat dilakukan pengujian gambar dan pengujian *webcam* secara *real-time*. Untuk pengambilan data deteksi objek digunakan pengujian gambar sedangkan untuk pengambilan data mengenai pengontrolan kecepatan kereta api digunakan pengujian *webcam* secara *real-time*. Pengujian data sampel dilakukan dengan berbagai variasi posisi, lebar celah *broken* dan ketinggian pasir. Contoh pengujian dapat dilihat pada Gambar 8.



a



b



c

Gambar 8. Contoh Uji Data gambar (a)*Broken*, (b)*Curved*, dan (c)*Erosion*

2.3. Perancangan Pengontrolan Kecepatan Motor Kereta Api

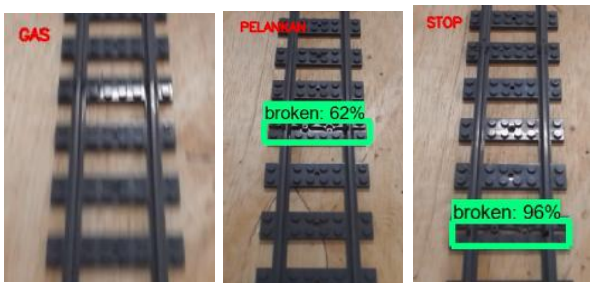
Perancangan pengontrolan kecepatan kereta api diawali dengan melakukan logika *if else* yang dilakukan pada *client*. Pengujian data yang mengeleuarkan output berupa nilai kepercayaan akan dijadikan pertimbangan untuk memberikan perintah. Perintah ini dibedakan dalam 3 kondisi berdasarkan nilai skor yang didapat. Skor tersebut merupakan nilai dari rentang 0 sampai 1. Logika *if else* dapat dilihat pada Gambar 9.

```
if any(scores_webcam[0] >= 0.7):
    gas=False
    warning=False
elif any(scores_webcam[0] >= 0.2):
    warning =True
    gas=False
else:
    gas=True
    warning=False
```

Gambar 9. Logika *If Else* dari *Client*

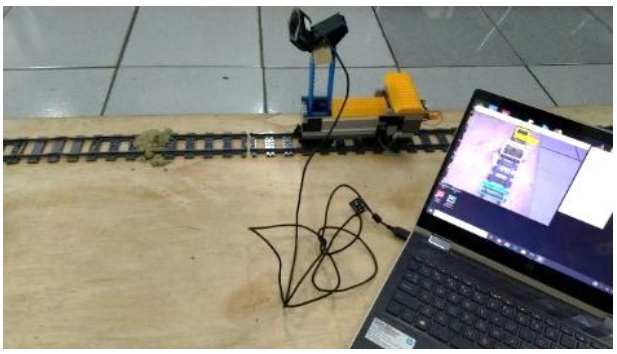
- Kondisi 1: Apabila tidak ada objek yang dideteksi atau dapat dikatakan tidak ada nilai skor maka *client* akan mengirimkan sinyal kontrol untuk memaksimumkan PWM. Sehingga kereta bergerak.
- Kondisi 2: Apabila ada objek yang dideteksi atau dapat dikatakan ada nilai skor namun berada pada nilai 0.2 – 0.7 maka *client* akan mengirimkan sinyal kontrol untuk menurunkan PWM. Sehingga kereta memperlambat.
- Kondisi 3: Apabila ada objek yang dideteksi atau dapat dikatakan ada nilai skor dan berada pada nilai diatas 0.7 maka *client* akan mengirimkan sinyal kontrol untuk meminimumkan PWM. Sehingga kereta berhenti.

Setelah melakukan pengkondisian berikut maka *client* akan mengirimkan perintah yang sudah dikonversikan menjadi tipe *integer* melalui protokol komunikasi TCP/IP ke mikrokontroler *NodeMCU*. Kondisi-kondisi tersebut kemudian diinisialkan besaran PWM yang akan dikeluarkan untuk menggerakkan motor. Untuk kondisi pertama diberi nilai PWM minimum, untuk kondisi ketiga diberi nilai PWM maksimum dan untuk kondisi kedua diberi nilai PWM diantaranya. Hasil logika dari konfisi-kondisi tersebut kemudian diimplementasikan langsung menggunakan *webcam* secara *real-time.* Hasil perintah pengontrolan dapat dilihat pada Gambar 10.



Gambar 10. Hasil Perintah Pengontrolan Kecepatan Motor Kereta api (a) Kondisi 1, (b) Kondisi 2, (c) Kondisi 3

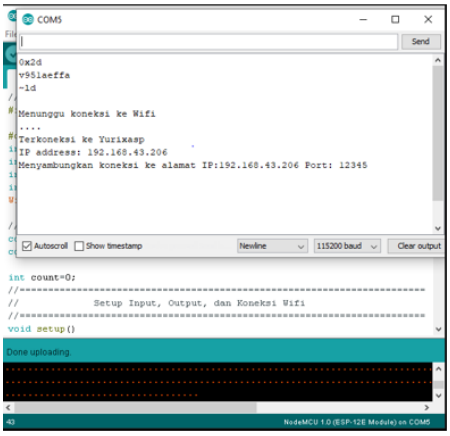
Dalam pengujiannya dilakukan perhitungan jarak kereta api terhadap objek pada saat kereta api berhenti dengan memberikan variasi nilai jarak kereta api terhadap sudut pandang kamera. *Set up* eksperimen dapat dilihat pada Gambar 11 dan cara pengukuran jarak dapat dilihat pada Gambar 12.



Gambar 11. *Set Up* Eksperimen

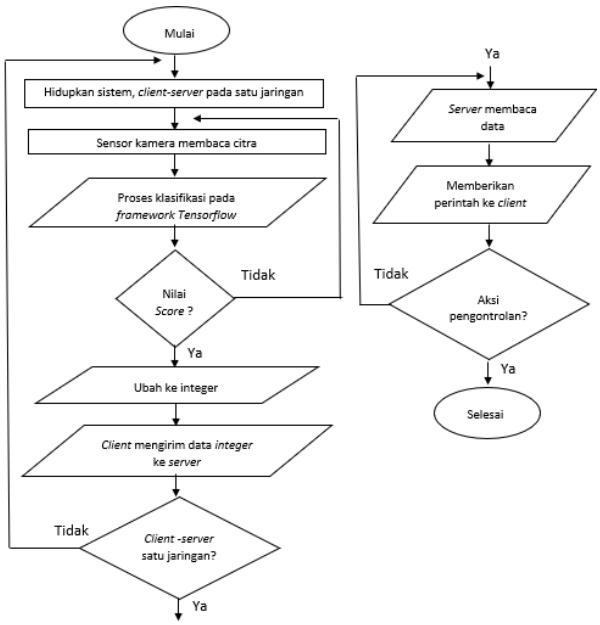
Gambar 12. Skema Pengukuran Jarak Kereta Api Terhadap Objek dan Sudut Pandang Kamera

2.4. Pengintegrasian Sistem



Gambar 13. Hasil Tampilan Komunikasi *Client* dan *Server*

Pengintegrasian sistem antara *client* dan *server* dilakukan melalui protokol komunikasi TCP/IP yang sudah *embedded* didalam *NodeMCU*. Agar *client* dan *server* dapat saling berkomunikasi maka keduanya harus berada pada satu jaringan yang sama. Kemudian pada *client* harus mengetahui alamat IP dan *port server* begitu juga pada *server* harus mengetahui alamat IP dan *port* pada *client*. Dalam penelitian ini IP *server* didapat 192.168.43.206. Hasil tampilan *client* dan *server* apabila sudah terhubung dapat dilihat pada Gambar 13 dan diagram alir pengintegrasian sistem dapat dilihat pada Gambar 14.



Gambar 14. Diagram Alir Pengintegrasian Sistem

3. Hasil dan Analisa

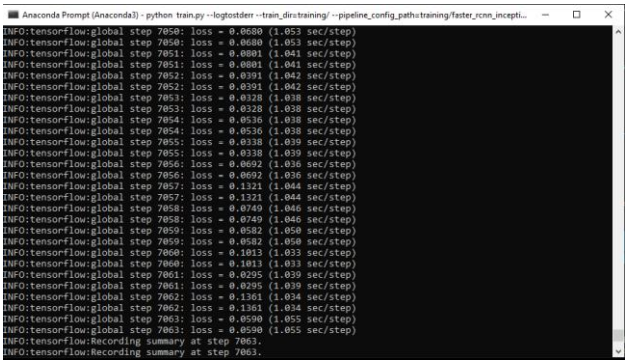
Perancangan metoda klasifikasi cacat pada rel dan deteksi penghalang kereta api menggunakan *deep learning* dilakukan sebagai salah satu alternatif dalam meningkatkan keamanan di kereta api. Penerapan pengontrolan pada *prototype* kereta api dapat diterapkan di dunia nyata seperti penerapan *camera vision* untuk kereta api inspeksi *drivereless* yang belum diterapkan di Indonesia. Pada penelitian ini telah dilakukan pengujian algoritma untuk pendeteksi objek dan pengujian pengontrolan kecepatan motor kereta api.

3.1. Hasil Pengujian Algoritma Pendeteksi Objek

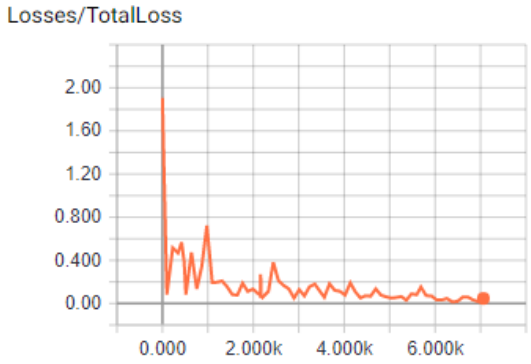
Pengujian algoritma pendeteksi objek telah dilakukan dengan melalui beberapa tahapan yaitu tahapan pelatihan dan tahapan pengambilan data uji.

3.1.1. Hasil Data Pelatihan

Data pelatihan dilakukan sebanyak 7063 *step* dengan waktu pelatihan kurang lebih 3 jam dan rata-rata waktu per *step* kurang dari 1 detik. Pemberhentian pelatihan dilakukan pada step ke 7063 karena nilai *loss* sudah sangat kecil yaitu sekitar dibawah 0.05. Hasil pelatihan dapat dilihat pada Gambar 15 dan grafik dari hasil pelatihan dapat dilihat pada Gambar 16.



Gambar 15. Tampilan Hasil Pelatihan pada *Command Prompt*

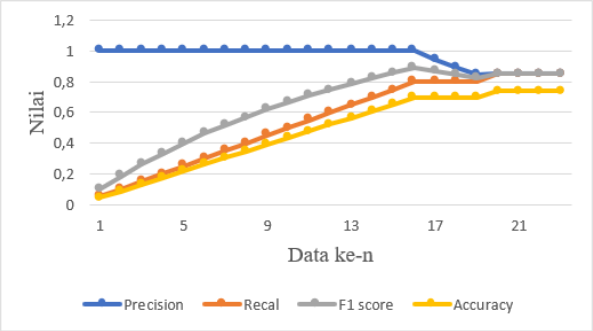


Gambar 16. Grafik Total *Loss* terhadap *Step* dari Hasil Pelatihan

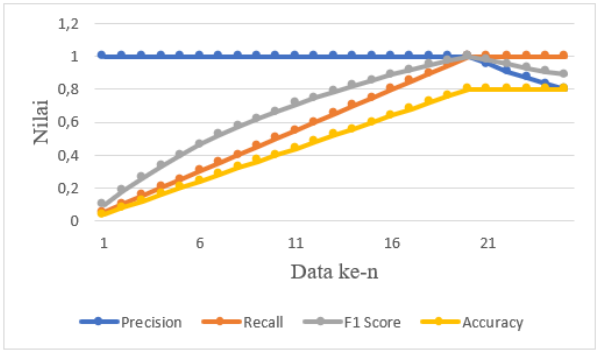
Grafik pada Gambar 16 merupakan visualisasi dari hasil pelatihan pada model yang digunakan. Terlihat bahwa semakin banyaknya jumlah pelatihan , nilai *loss function* semakin konvergen menuju nol. Artinya bahwa nilai *error* yang dihasilkan sudah cukup kecil.

3.1.2. Hasil Data Pengujian

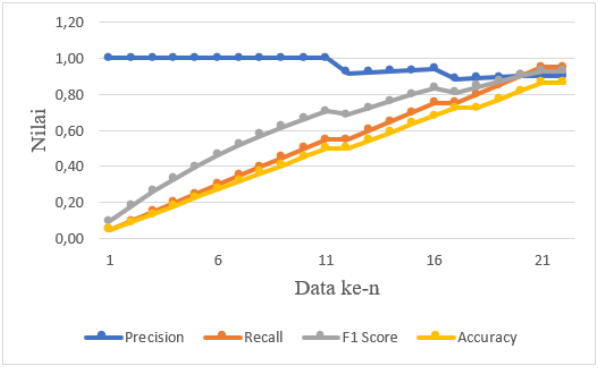
Pengujian dilakukan pada masing-masing kelas. Kemudian dihitung nilai performansi berdasarkan *confusion matrix*. Performansi yang dihitung yaitu nilai *precision, recall, F1 score* dan *accuracy*. Grafik performansi dari masing – masing label dapat dilihat pada Gambar 17, Gambar 18 dan Gambar 19.



Gambar 17. Grafik Performansi pada Label *Broken*



Gambar 18. Grafik Performansi pada Label *Curved*



Gambar 19. Grafik Performansi pada Label *Erosion*

Dari ketiga grafik diatas dapat terlihat bahwa nilai *precision, recall, F1-score, dan accuracy* cenderung meningkat secara konsisten dan bergerak saling mendekati hingga mendekati nilai 1 seiring bertambahnya data pengujian. Hal ini dapat dikatakan bahwa performa dari model yang digunakan sudah cukup baik karena kondisi ini menunjukkan bahwa model mampu mencapai keseimbangan antara ketepatan prediksi (*precision)* dan kelengkapan deteksi (*recall*) pada setiap kelas. Nilai *F1-score* yang relatif tinggi menandakan bahwa kesalahan berupa *false positive* dan *false negative* berada pada tingkat yang rendah dan tidak didominasi oleh salah satu jenis kesalahan tertentu. Nilai dari pengukuran performansi pada masing-masing label disajikan pada Tabel 1.

Tabel 1. Nilai-Nilai Performansi pada Masing-Masing Label

Kelas/Label	Total			*Precision*	*Recall*	*F1 Score*	*Accuracy*
	TP	FP	FN				
Broken	17	3	3	0,85	0,85	0,85	0,74
Curved	20	5	-	0,8	1,00	0,88	0,8
Erosion	19	2	1	0,90	0,95	0,93	0,86

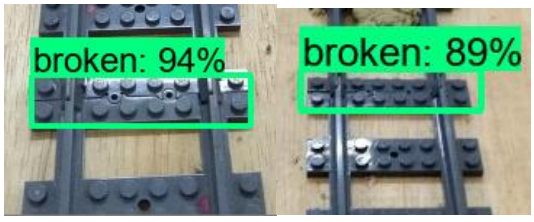
Note: TP (true positive), FP (false positive), FN (false negative)

Pada tabel diperlihatkan bahwa kelas *curved* menunjukkan nilai *recall* sempurna (1,00), namun *precision* lebih rendah (0,80). Kondisi ini mengindikasikan bahwa seluruh objek rel melengkung (*curved*) yang benar-benar ada pada data uji berhasil terdeteksi oleh model, sehingga tidak terdapat *false*

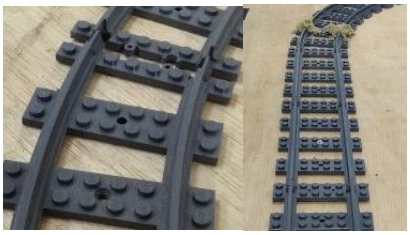
negative pada kelas ini. Dengan kata lain, sistem memiliki kelengkapan deteksi yang sangat baik untuk kelas *curved*. Namun demikian, nilai *precision* yang lebih rendah menunjukkan adanya *false positive*, yaitu beberapa objek yang sebenarnya bukan *curved* diklasifikasikan sebagai *curved*. Hal ini mengindikasikan bahwa model cenderung lebih agresif atau sensitif dalam mendeteksi kelas *curved*. Secara visual, karakteristik kelengkungan rel merupakan pola global yang relatif mudah dikenali oleh model berbasis *CNN*, sehingga *backbone Inception V2* mampu mengekstraksi fitur bentuk secara efektif. Akan tetapi, sensitivitas yang tinggi ini juga menyebabkan model salah menginterpretasikan kondisi tertentu seperti distorsi perspektif kamera, sudut pengambilan citra, atau bayangan rel sebagai kelengkungan rel. Fenomena ini menunjukkan adanya *trade-off* antara *recall* dan *precision*. Pada kelas *curved*, model dioptimalkan untuk meminimalkan risiko objek *curved* yang terlewat (*false negative*), sehingga *recall* menjadi sangat tinggi. Konsekuensinya, sistem menerima peningkatan *false positive* yang berdampak pada penurunan *precision*.

Selanjutnya, *false positive* pada kelas *broken* terjadi ketika sistem mengklasifikasikan rel sebagai *broken* meskipun kondisi tersebut sebenarnya bukan cacat. Hal ini dikarenakan data *training* pada broken memiliki variasi lebar celah yang sangat kecil dengan nilai minimum kisaran 0,1 cm sehingga model masih kesulitan untuk membedakan mana yang celah dan mana yang bukan celah dalam rentang nilai yang sangat kecil. Sedangkan untuk data-data *false negative* terjadi pada label *erosion* dan juga pada label *broken* yang seharusnya terdeteksi tetapi tidak terdeteksi. Hal ini dikarenakan untuk label *broken* kurangnya varian data pada saat rel berbelok sehingga sistem sulit membedakan. Sedangkan untuk label *erosion* juga dikarenakan kurangnya varian data posisi pengambilan. Sehingga dari kejauhan sistem masih belum mengenali. Akan tetapi ini bukan menjadi masalah yang krusial dikarenakan dalam pengaplikasian dikereta api memiliki jarak pandang yang sudah ditentukan. Contoh gambar *false positive* dan *false negative* diperlihatkan pada Gambar 20 dan Gambar 21.

Berdasarkan hasil, rata-rata nilai akurasi yang diperoleh ialah 80%, lebih rendah dibandingkan penelitian sebelumnya (Merve,2024) yang memperlihatkan akurasi di atas 90%. Hal ini dikarenakan ukuran dataset relatif kecil, kompleksitas visual objek yang tinggi serta keterbatasan sumber daya komputasi.



Gambar 20. Hasil *False Positive* untuk Label *Broken*



Gambar 21. Hasil *False Negative* untuk label *Erosion*

3.2. Hasil Pengontrolan Motor Kereta Api

Setelah dilakukan pendeteksian objek maka dilakukan pengujian kecepatan motor kereta api dengan mengintegrasikan *client server* melalui protokol TCP/IP. Pengujian pengontrolan kecepatan motor kereta api dilakukan pada saat *real-time* dengan menetapkan jarak antara kereta api terhadap sudut kemiringan sensor kamera sebesar 2.5 cm. Dalam pengujiannya dilakukan pengaturan kecepatan motor kereta api berdasarkan kondisi-kondisi yang sudah ditentukan. Pengujian dilakukan pada masing-masing label dan dihitung jarak kereta api terhadap objek yang dideteksi saat berhenti. Contoh pengujian pengontrolan dapat dilihat pada Gambar 22.

a b c

Gambar 22. Hasil Uji Pengontrolan saat *Real-Time* (a) Kondisi 1, (b) Kondisi 2, (c) Kondisi 3

Pada Gambar 22 dapat dilihat bahwa pada saat objek belum terdeteksi (Gb.(a)), sistem akan memberikan sinyal kontrol berupa PWM maksimum dan akan menampilkan teks bertuliskan “GAS”. Kemudian pada saat sistem mendeteksi objek namun tingkat kepercayaan berada diantara 20% - 70% (Gb.(b)) maka sinyal kontrol akan memberikan perintah untuk menurunkan PWM dan menampilkan teks bertuliskan “PELANKAN”. Lalu pada saat sistem mendeteksi objek dan tingkat kepercayaan berada di atas 70% (Gb.(c)) maka sinyal kontrol akan memberikan perintah berupa PWM minimum serta menampilkan teks bertuliskan “STOP”. Saat pengujian dilakukan didapatkan nilai minimum lebar celah rel terputus pada sambungan yang mampu dideteksi sensor kamera pada saat *real-time* berkisar 1 cm. Sedangkan nilai minimum ketinggian pasir yang mampu dideteksi sensor kamera pada saat *real-time* berkisar 0.5 cm diatas bantalan rel.

Setelah itu dilakukan penghitungan nilai jarak antara kereta api terhadap objek yang dideteksi saat berhenti pada masing-masing label dengan melakukan variasi jarak kereta api terhadap sudut pandanga kamera. Variasi ini dilakukan dengan nilai S_o = 2,5 cm , 5cm dan 7,5 cm. Dengan menggunakan nilai lebar celah 1 cm dan nilai ketinggian pasir 0,5 cm diatas bantalan rel. Pengambilan data dilakukan sebanyak 10 kali pengukuran pada masing-masing label. Hasil pengujian pertama untuk nilai S_0 = 2,5 cm dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian Jarak antar Kereta Api terhadap Objek ketika Berhenti saati S_0 = 2,5 cm

Data-n	S_t (cm) *Broken*	S_t (cm) *Curved*	S_t (cm) *Erosion*
1	6,3	10,2	9,7
2	13	6,7	5,5
3	15,1	9	6,5
4	12	11,7	7
5	13,2	6,4	13
6	14,5	6,8	10,5
7	15,6	7,7	7,8
8	11,9	5,5	11
9	15,5	6,5	14
10	15,3	9,7	7
Rata-rata	13,24	8,02	9,2

Selanjutnya, hasil pengujian saat jarak sudut pandang kamera (S_0) diatur sebesar 5 cm ditampilkan pada Tabel 3. Sedangkan untuk pengujian dengan S_0 sebesar 7,5 cm ditunjukkan pada Tabel 4. Adapun rangkuman rata-rata keseluruhan jarak kereta terhadap objek saat berhenti dari ketiga pengujian tersebut dapat dilihat pada Tabel 5.

Tabel 3. Hasil Pengujian Jarak antar Kereta Api terhadap Objek ketika Berhenti saati S_0 = 5 cm

Data-n	S_t (cm) *Broken*	S_t (cm) *Curved*	S_t (cm) *Erosion*
1	21	24,7	18
2	8	26,4	21,6
3	12,5	25,2	11,7
4	14	17	22,2
5	20,5	22,5	24,2
6	13,3	20	14,1
7	13	14	15,6
8	10,2	13,6	21
9	21,1	22	17
10	7,5	22,2	18,5
Rata2	14,11	20,76	18,39

Tabel 4. Hasil Pengujian Jarak antar Kereta Api terhadap Objek ketika Berhenti saati S_0 = 7,5 cm

Data-n	S_t (cm) *Broken*	S_t (cm) *Curved*	S_t (cm) *Erosion*
1	11,5	26,2	19,5
2	9,5	27,9	23,1
3	24,7	26,7	13,2
4	17	18,5	23,7
5	15,1	24	25,7
6	20,7	21,5	15,6
7	27,7	15,5	17,1
8	12,3	15,1	22,5
9	27,7	23,5	18,5
10	22,6	23,7	20
Rata2	18,88	22,26	19,89

Tabel 5. Hasil Keseluruhan Rata-Rata Jarak antar Kereta Api terhadap Objek saat Berhenti

S_o (cm)	Rata-rata S_t (cm) *broken*	Rata-rata S_t (cm) *curved*	Rata-rata S_t (cm) *erosion*	Rata-rata S_t (cm) keseluruhan
2,5	13,24	8,02	9,2	10,15
5	14,11	20,76	18,39	17,75
7,5	18,88	22,26	19,89	20,34

Hasil pengujian pada Tabel 5, menunjukkan bahwa rata-rata jarak pemberhentian (S_t) meningkat seiring bertambahnya S_o. Pada nilai S_o yang lebih kecil, kamera berada lebih dekat dengan objek dan sudut pandang relatif tajam, sehingga objek tampil lebih besar pada citra dan memiliki kontras yang lebih jelas terhadap latar rel. Dengan demikian, nilai *confidence* juga meningkat lebih cepat dan sistem memicu pengereman lebih awal.

Pada nilai S_o yang lebih besar, objek akan terlihat lebih awal dari pandangan kamera, walaupun representasi visual objek cenderung lebih kecil dan kurang kontras, model dapat mampu mengenali objek dari kejauhan sehingga nilai *confidence* juga sama-sama meningkat lebih cepat dan sistem memicu pengereman lebih awal.

Berdasarkan hasil ini, dapat dikatakan bahwa semakin besar nilai S_0, maka jarak kereta saat berhenti terhadap objek juga semakin besar. Hal ini terjadi karena kamera dapat mendeteksi objek lebih awal saat diletakkan lebih jauh, sehingga sistem memberikan perintah berhenti lebih cepat.

4. Kesimpulan

Berdasarkan hasil penelitian, dapat disimpulkan bahwa sistem untuk mendeteksi kerusakan dan hambatan pada rel kereta api menggunakan metode *deep learning* dengan algoritma *Faster R-CNN* telah berhasil dikembangkan. Pengujian menunjukkan bahwa sistem memiliki performa yang baik, dengan nilai rata-rata *precision* sebesar 0,85, *recall* 0,93, *accuracy* 0,80, *dan F1 Score* 0,88. Angka-angka ini menunjukkan bahwa model mampu mengidentifikasi objek dengan cukup akurat karena nilainya mendekati 1. Selain itu, sistem pengontrol kecepatan pada motor kereta juga berhasil diimplementasikan secara *real-time*, di mana kecepatan kereta disesuaikan secara otomatis berdasarkan tingkat kepercayaan dari hasil klasifikasi. Pengujian lebih lanjut menunjukkan bahwa jarak antara kereta dan objek saat berhenti dipengaruhi oleh posisi kamera terhadap rel, khususnya sudut pandang kamera.

Kontribusi utama penelitian ini terletak pada demonstrasi integrasi *end-to-end* antara algoritma *Faster R-CNN* dan sistem kendali kecepatan kereta secara real-time, yang merupakan pengembangan lebih lanjut dibandingkan penelitian sebelumnya yang umumnya hanya berfokus pada aspek deteksi. Namun demikian, purwarupa yang

dikembangkan masih memiliki keterbatasan, terutama pada ukuran dataset yang relatif kecil serta pengujian yang dilakukan pada lingkungan laboratorium yang disederhanakan. Oleh karena itu, penelitian selanjutnya diarahkan pada pengujian menggunakan dataset yang lebih besar dan beragam, serta eksplorasi arsitektur *deep learning* yang lebih ringan dan efisien untuk implementasi pada perangkat *embedded*.

Referensi

[1]. H. Dwiatmoko, M. Isradi, J. Prasetijo, and A. Hamid, "Comparative study of passenger satisfaction with regional rail transport in Indonesia and Malaysia," *European Journal of Scientific Innovation and Technology*, vol. 2, no. 2, pp. 32–40, Apr. 2022.

[2]. J. Sresakoolchai and S. Kaewunruen, "Railway infrastructure maintenance efficiency improvement using deep reinforcement learning integrated with digital twin based on track geometry and component defects," *Scientific Reports*, vol. 13, no. 1, 2023, doi: 10.1038/s41598-023-29526-8.

[3]. V. N. A. Putri, K. Usman, I. Kustiani, dan A. M. Siregar, "Analisis tingkat risiko penyebab rel patah pada jalur kereta api wilayah Divre IV Tanjung Karang," *Jurnal Teknik Sipil dan Lingkungan*, vol. 9, no. 2, pp. 283–292, Oct. 2024.

[4]. R. Jamieson and D. Smilek, "Understanding the role of human attention in railway inspection errors," in *Proceedings of the Wheel/Rail Interaction Conference*, 2022.

[5]. W. Gong, M. F. Akbar, G. N. Jawad, M. F. P. Mohamed, and M. N. A. Wahab, "Nondestructive testing technologies for rail inspection: A review," *Coatings*, vol. 12, no. 11, art. no. 1790, Nov. 2022.

[6]. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[7]. M. Krichen, "Convolutional neural networks: A survey," *Computers*, vol. 12, no. 8, art. no. 151, Jul. 2023, doi: 10.3390/computers12080151.

[8]. J. Pardede dan H. Hardiansah, "Deteksi objek kereta api menggunakan metode Faster R-CNN dengan arsitektur VGG16," *Jurnal Ilmiah Ilmu Komputer*, vol. 7, no. 1, 2022.

[9]. E. Indasyah, F. Ibrahim, D. F. Syahbana, and F. Istiqomah, "Automated visual inspection system of gear surface defects detection using Faster R-CNN," in *Proceedings of the 2023 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, Surabaya, Indonesia, 2023, pp. 899–904, doi: 10.1109/ICAMIMIA60881.2023.10427945.

[10]. M. Yilmazer and M. Karakose, "Fastener and rail surface defects detection with deep learning techniques," *International Journal of Advances in Intelligent Informatics*, vol. 10, no. 2, pp. 253–264, May 2024, doi: 10.26555/ijain.v10i2.1237.

[11]. X. Chen and H. Zhang, "Rail surface defects detection based on Faster R-CNN," in *Proceedings of the 2020 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, Vientiane, Laos, 2020, pp. 228–231, doi: 10.1109/ICITBS49701.2020.00061.

[12]. Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013, doi: 10.1109/TPAMI.2013.50.

[13]. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[14]. Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.

[15]. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Procedding Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 91–99.

[16]. R. Indrakusuma, N. El Hafizah, dan D. R. A. Ardi, "Decision support to improve railway track maintenance in Indonesia: A life cycle cost approach," *Jurnal Teknologi dan Manajemen*, vol. 5, no. 2, pp. 162–177, 2024, doi: 10.31284/j.jtm.2024.v5i2.5945.

[17]. S. Mohammadi, S. S. Karganroudi, M. Adda, and H. Ibrahim, "Rail defect classification with deep learning method," *Green Energy and Intelligent Transportation*, vol. 3, art. no. 100332, 2025, doi: 10.1016/j.geits.2025.100332.

[18]. F. Yan, Y. Gu, and Y. Sun, "Deep learning-based train obstacle detection technology: Application and testing in metros," *Electronics*, vol. 14, no. 7, 2025, doi: 10.3390/electronics14071318.