IMPLEMENTASI METODE *WAVELET* DAN *BACKPROPAGATION NEURAL NETWORK* PADA DETEKSI RETAK JALAN RAYA BERBASIS PENGOLAHAN CITRA

SKRIPSI

Disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik (S.T)

Disusun Oleh:

TEGAR PRIYO UTOMO

NPM. 3332160055

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SULTAN AGENG TIRTAYASA

2023

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya sebagai penulis skripsi berikut:

Judul	:	Implementasi Metode Wavelet Dan Backpropagation Neural Network Pada Deteksi Retak Jalan Raya Berbasis Pengolahan Citra
Nama Mahasiswa	:	Tegar Priyo Utomo
NPM	:	3332160055
Fakultas/Jurusan	:	Teknik/Teknik Elektro

Menyatakan dengan sesunguhnya bahwa Skripsi tersebut di atas adalah benar-benar hasil karya asli saya dan tidak memuat hasil karya orang lain, kecuali dinyatakan melalui rujukan yang benar dan dapat dipertanggung jawabkan. Apabila dikemudian hari ditemukan hal-hal yang menunjukkan bahwa sebagian atau seluruh karya ini bukan karya saya, maka saya bersedia dituntut melalui hukum yang berlaku. Saya juga bersedia menanggung segala akibat hukum yang timbul dari pernyataan yang secara sadar dan sengaja saya nyatakan melalui lembar ini

Cilegon, Agustus 2023

Tegar Priyo Utomo

3332160038

LEMBAR PENGESAHAN

Dengan ini saya sebagai penulis Skripsi berikut:

Judul : Implementasi Metode Wavelet dan *Backpropagation Neural Network* Pada Deteksi Retak Jalan Raya Berbasis Pengolahan Citra

Nama Mahasiswa : Tegar Priyo Utomo

NPM : 3332160055

Fakultas/Jurusan : TeknikTeknik Elektro

Telah diuji dan dipertahankan pada tanggal 10 Juni 2022 melalui Sidang Skripsi di Fakultas Teknik Universitas Sultan Ageng Tirtayasa Cilegon dan dinyatakan **LULUS**.

Dewan Penguji

		Tanda Tangan
Pembimbing I	: Rian Fahrizal, S.T., M.Eng.	.....
Pembimbing II	: Dr. Eng. Rocky Alfanz, M.Sc	.....
Penguji I	: Ir. Ri Munarto, M.Eng.	.....
Penguji II	: Dr. Siswo Wardoyo, M.Eng.	.....

Mengetahui,

Ketua Jurusan

Dr. Romi Wiryadinata, S.T., M.Eng.

NIP. 198307032009121006

PRAKATA

Puji syukur saya panjatkan kepada Allah Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Sultan Ageng Tirtayasa. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1. Kedua orang tua ibu dan bapak tercinta, dan adik-adik saya yang telah memberikan nasehat, semangat, doa, dan materi yang tak terhingga nilainya.
2. Bapak Dr. Romi Wiryadinata, S.T., M.Eng. selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Untirta.
3. Bapak Rian Fahrizal, S.T., M.Eng. selaku pembimbing I yang telah meluangkan waktunya dengan sabar memberikan bimbingan selama penyusunan skripsi ini.
4. Bapak Dr. Eng. Rocky Alfanz, M.Sc.selaku pembimbing II yang telah meluangkan waktunya dan dengan sabar memberikan bimbingan selama penyusunan skripsi ini.
5. Seluruh teman-teman jurusan Teknik Elektro dan BIDIKMISI 2016 yang telah mendukung dan membantu sejak awal masa perkuliahan hingga sekarang.

Akhir kata, saya berharap Tuhan Yang Maha Esa membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Cilegon, Agustus 2023

Tegar Priyo Utoomo
3332160055

ABSTRAK

Tegar Priyo Utomo

Teknik Elektro

Implementasi Metode Wavelet Dan Backpropagation Neural Network Pada Deteksi Retak Jalan Raya Berbasis Pengolahan Citra

Salah satu metode yang dapat digunakan dalam pengolahan citra untuk deteksi retak adalah CNN. Jumlah data citra yang digunakan pada penelitian ini yaitu 876 citra yang terdiri dari 560 data pelatihan, 140 data validasi, dan 176 data pengujian. Arsitektur model CNN yang digunakan terdiri dari tiga lapisan konvolusi dan tiga lapisan *max pooling*. Tahap pelatihan, model dikenalkan dengan pola citra retak buaya, retak garis dan tidak retak kemudian divalidasi. Tahap pengujian, model mengklasifikasikan citra retak buaya, retak garis dan tidak retak. Tingkat akurasi model pada tahap pelatihan sebesar 96,43% dan pada tahap pengujian sebesar 96,65%.

Kata kunci: *deep learning*, CNN, DWT, retak

ABSTRACT

Tegar Priyo Utomo

Electrical Engineering

Implementation of The Wavelet Method and Backpropagation Neural Network on Road Crack Detection Based on Image Processing

One method that can be used in image processing for *crack* detection is CNN. The amount of image data used in this study is 876 images consisting of 560 *training* data, 140 validation data, and 176 testing data. The CNN model architecture used consists of three convolution *layer*s and three max *pooling layer*s. In the *training* stage, the model is introduced to the image patterns of crocodile *crack*s, line *crack*s and no *crack*s and then validated. In the testing stage, the model classifies crocodile *crack*, line *crack* and no *crack* images. The accuracy of the model in the *training* stage is 96,43% and in the testing stage is 96,65%.

Keywords: deep learning, CNN, DWT, *crack*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pemerintah di seluruh dunia selalu menjadikan kualitas perkerasan jalan sebagai perhatian utama [1]. Pemeriksaan kondisi jalan secara berkala merupakan hal penting yang digunakan untuk mengumpulkan data yang dibutuhkan dalam perencanaan dan pembuatan jadwal tindakan pemeliharaan dan perbaikan. Retak adalah kerusakan jalan yang paling umum ditemukan selama pemeriksaan tersebut [2]. Kerusakan berbentuk cacat permukaan dan retakan secara konsisten teramati di seluruh wilayah yang diteliti, dan memiliki proporsi tingkat kerusakan yang signifikan jika dibandingkan dengan variasi jenis kerusakan lainnya. Pendapat ini juga diperkuat oleh pandangan para ahli yang menganggap bahwa cacat permukaan dan retakan adalah elemen dominan dalam kerusakan perkerasan jalan di ruas-ruas jalan nasional dan provinsi di Indonesia [3].

Pengecekan kondisi jalan oleh petugas lapangan masih dilakukan secara manual yaitu pengecekan dengan indra penglihatan [4]. Cara ini memerlukan banyak waktu dan tidak efektif dan mempunyai resiko bahaya ditengah lalu lintas jalanan yang padat. Dibutuhkan sebuah metode baru yang lebih aman dan dapat memudahkan petugas lapangan untuk memantau kondisi jalan guna keperluan perawatan dan pemeliharaan jalan. Teknologi *computer vision* dan *machine learning* telah berhasil diterapkan sebagai solusi untuk mengklasifikasi kerusakan jalan [5-9]. Metode pengolahan citra digunakan untuk mendapatkan informasi berupa ekstrasi ciri dari data citra kerusakan jalan [10]. Penelitian lain menggunakan metode pengolahan citra untuk mendapatkan ekstraksi ciri yaitu, *Local Binary Patern* (LBP) [11], *Gray Level Co-Occurrence Matrix* (GLCM)[11], *Hue Saturation Value* (HSV) [12] dan untuk metode klasifikasi antara lain: *K-Nearest Neighbor* (K-NN) [13], LDA [14], *Support Vector Machine* (SVM) [15], dan *Neural Network* [16].

Penelitian ini mengusulkan metode untuk mendeteksi retakan jalan, yaitu menggunakan pengolahan citra. Selain metode ini telah banyak diimplementasikan

untuk mendeteksi retak jalan karena biayanya lebih sedikit dibandingkan dengan inspeksi manual [17], Metode ini juga mengurangi gangguan terhadap lalu lintas umum dan bahaya kepadaatan lalu lintas jalan bagi petugas lapangan selama melakukan survei [18]. Bidang *computer vision* yang telah mencapai kemajuan besar adalah *deep learning*. *Deep learning* merupakan implementasi dari *neural network* yang memiliki banyak *hidden layers* [19]. *Deep learning* mendapatkan hasil yang lebih baik untuk mendeteksi retak jalan, dalam penelitian ini metode *CNN* dikombinasikan dengan transformasi Wavelet sebagai metode ekstraksi ciri untuk mendapatkan hasil yang lebih baik.

Deteksi retak jalan secara *real-time* pada penelitian ini menggunakan metode YOLO yang lebih unggul baik dari segi akurasi dan kecepatan [20][21]. Meskipun memiliki berbagai versi, YOLOv4 adalah salah satu yang terbaru, mencapai rata-rata 10% lebih banyak presisi (AP) pada *dataset* MS COCO dibandingkan versi sebelumnya, YOLOv3 [20]. Algoritma YOLOv4 digunakan untuk mendeteksi retak jalan secara *real-time* pada penelitian ini.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijabarkan, maka rumusan masalah untuk penelitian ini meliputi:

1. Bagaimana merancang sistem deteksi retak jalan raya menggunakan CNN dan YOLO?
2. Bagaimana pengaruh parameter *epoch* dan *learning rate* dalam proses pelatihan data menggunakan metode CNN?
3. Bagaimana membandingkan hasil kinerja CNN, GUI, dan YOLO berupa *accuracy, precision, recall, f1score* pada deteksi retak jalan raya?

1.3 Tujuan Penelitian

Berdasarkan latar belakang masalah dan rumusan masalah yang telah dijabarkan, maka tujuan penelitian yang ingin dicapai diantaranya yaitu:

1. Membangun sistem yang dapat mendeteksi retak pada jalan raya menggunakan metode CNN dan YOLO.

2. Mengetahui pengaruh parameter *epoch* dan *learning rate* dalam proses *training* data menggunakan metode CNN.
3. Membandingkan hasil kinerja CNN, GUI, dan YOLO berupa *accuracy, precision, recall,* F1-*score* pada deteksi retak jalan raya.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Hasil penelitian diharapkan dapat bermanfaat untuk membantu mempermudah dinas terkait dalam pengecekan kondisi jalan.
2. Menambah referensi untuk pengaplikasian pengolahan citra untuk ekstrasi ciri dan klasifikasi retak pada jalan raya.
3. Memberikan dukungan teknologi terkait dalam pemeriksaan dan pemeliharaan jalan yang efektif dan efisien.

1.5 Batasan Masalah

Batasan masalah yang terdapat pada penelitian ini diuraikan sebagai berikut:

1. Bahasa pemprograman yang digunakan adalah Python.
2. Ekstraksi ciri pada penelitian ini menggunakan metode Wavelet.
3. Metode klasifikasi yang digunakan pada penelitian ini adalah metode CNN.
4. Kerusakan jalan yang dibahas hanya retakan pada jalan jenis aspal.
5. Analisis berfokus pada klasifikasi retak garis, retak buaya dan tidak retak.
6. Penelitian ini hanya fokus pada analisa dan pengembangan *software* tidak mencakup pengembangan *hardware*.

1.6 Sistematika Penulisan

Penyusunan penulisan skripsi ini dibagi dalam 5 bab, sehingga isi dari masing-masing bab meliputi sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas mengenai latar belakang masalah, perumusan masalah, batasan masalah, tujuan, serta sistematika penulisan pada penelitian.

BAB II TINJAUAN PUSTAKA

Bab ini memuat mengenai teori pengolahan citra pada pengolahan citra, metode ekstraksi ciri Wavelet dan metode CNN.

BAB III METODOLOGI PENELITIAN

Bab ini memuat mengenai metodologi penelitian yang digunakan meliputi pengolahan citra pada citra dan perancangan penelitian dengan menggunakan metode Wavelet, CNN, dan YOLOv4.

BAB IV HASIL DAN PEMBAHASAN

Bab ini memuat hasil data dari ekstraksi citra untuk program yang telah dibuat, serta pembahasan dari klasifikasi hasil ekstrasi ciri menggunakan Wavelet dan CNN.

BAB V PENUTUP

Bab ini membahas mengenai kesimpulan dari penelitian yang telah dilakukan secara singkat, padat, dan jelas. Bab ini juga membahas mengenai saran untuk penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Retak Jalan Raya

Jalan merujuk kepada semua komponen jalan, termasuk infrastruktur tambahan dan fasilitas pendukungnya yang digunakan oleh masyarakat umum untuk transportasi, dan dapat berlokasi pada berbagai tingkat, mulai dari permukaan tanah, hingga kedalaman bawah tanah dan air, serta di atas permukaan air. Ini tidak termasuk jalur kereta api dan jalur kabel [22]. Secara umum, kerusakan pada jalan dapat digolongkan menjadi dua kategori, yaitu kerusakan yang bersifat struktural dan kerusakan yang bersifat fungsional. Kerusakan struktural melibatkan kegagalan pada struktur perkerasan atau kerusakan pada beberapa komponen perkerasan. Jenis kerusakan ini mengakibatkan perkerasan tidak lagi mampu menahan beban dari kendaraan yang melintas. Sementara itu, kerusakan fungsional merupakan jenis kerusakan yang mempengaruhi aspek keamanan dan kenyamanan para pengguna jalan. Beberapa contoh kerusakan jalan meliputi retak-retak pada permukaan jalan dan gangguan pada marka jalan. [23].

Retak kulit buaya adalah jenis keretakan yang menyerupai pola kulit buaya. Kerusakan ini mencakup berbagai istilah seperti retak kulit buaya, *chickenwire cracks, alligator cracks, polygonal cracks,* dan *crazing*. Retakan ini memiliki lebar minimal sekitar 3 mm dan biasanya terhubung membentuk serangkaian pola kotak-kotak kecil yang mirip dengan kulit buaya atau jaring ayam. Wilayah di mana retak kulit buaya terjadi dapat bervariasi dalam ukuran, dengan beberapa daerah yang memiliki kerusakan tersebut cenderung lebih luas daripada yang lain. Hal ini disebabkan oleh beban lalu lintas berulang yang melebihi kapasitas lapisan permukaan untuk menahan beban tersebut.

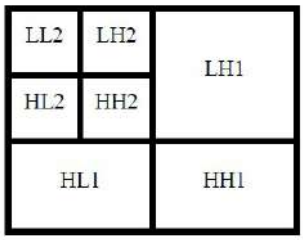
Retak garis adalah jenis keretakan yang berbentuk garis dan dapat terjadi dalam arah memanjang, melintang, atau diagonal. Beberapa jenis kerusakan retak yang termasuk dalam kategori ini meliputi retak di tepi perkerasan, retak di pertemuan antara perkerasan dan bahu jalan, retak di sambungan antara segmen jalan, dan retak di

sambungan saat pelebaran jalan. Biasanya, retak garis mengacu pada keretakan yang terletak di sisi tepi perkerasan dekat bahu jalan dan memiliki bentuk garis yang membentang, kadang-kadang dengan cabang-cabang yang mengarah ke bahu. Retak ini bisa terdiri dari beberapa celah yang berjajar sejajar..

2.2 Transformasi Wavelet

Ekstraksi fitur adalah tahap di mana citra mengalami perubahan agar dapat mengungkapkan informasi yang tersembunyi dengan lebih jelas. Transformasi citra, dalam konteks ini, merujuk pada perubahan citra dari satu domain ke domain lainnya. Melalui langkah transformasi ini, citra dapat dijelaskan sebagai hasil kombinasi linear dari sinyal dasar yang biasanya disebut sebagai fungsi dasar [24].

Wavelet merupakan gelombang singkat atau gelombang kecil. Transformasi Wavelet adalah proses yang mengubah suatu sinyal menjadi rangkaian gelombang-gelombang singkat. Gelombang-gelombang singkat ini adalah fungsi yang memiliki lokasi pada waktu yang berbeda. Transformasi Wavelet memiliki kapasitas untuk mengenali informasi tentang frekuensi yang terkandung dalam sinyal dan juga menawarkan wawasan tentang skala, durasi, atau waktu. Wavelet digunakan untuk memeriksa karakteristik bentuk sinyal gelombang yang timbul dari perpaduan antara faktor waktu atau skala serta faktor frekuensi. Berikut tahapan proses dan hasil dari transformasi Wavelet pada ilustrasi yang ada di Gambar 2.1.



Gambar 2.1 Dekomposisi Wavelet

Gambar 2.1 menggambarkan proses dekomposisi citra setelah menjalani transformasi Wavelet. Dalam fase ini, citra awal dibagi menjadi empat sub-citra baru yang akan menggantikannya. Masing-masing sub-citra ini memiliki ukuran yang merupakan sepertiga dari citra asli. Tiga sub-citra yang terletak di sudut kanan atas,

sudut kanan bawah, dan sudut kiri bawah akan menampilkan tingkat detail yang lebih tinggi dibandingkan dengan citra aslinya karena mereka mengandung komponen frekuensi tinggi dari citra asli. Sebaliknya, sub-citra yang berada di sudut kiri atas akan lebih menyerupai citra asli dan lebih halus karena mengandung komponen frekuensi rendah dari citra asli tersebut. Selanjutnya, sub-citra di sudut kiri atas yang berisi frekuensi rendah juga dapat dibagi menjadi empat sub-citra baru. Proses ini dapat diulangi sesuai dengan tingkat transformasi yang diterapkan.

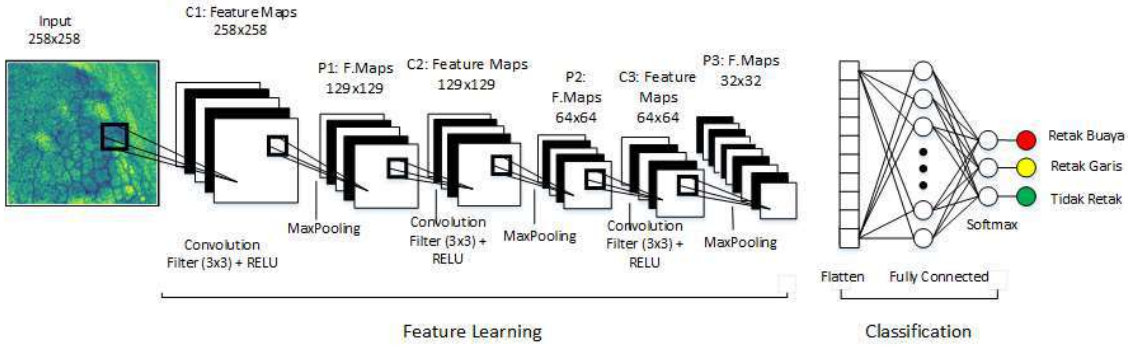
Dalam situasi citra dua dimensi, terdapat dua metode yang digunakan untuk mengubah atau memisahkan nilai-nilai piksel, yakni pendekatan standar dan pendekatan yang tidak biasa. Kedua metode ini berdasarkan pada transformasi Wavelet 1 dimensi. Pendekatan dekomposisi standar melibatkan penerapan transformasi Wavelet 1 dimensi terlebih dahulu pada setiap baris citra, dan kemudian pada setiap kolom citra. Dengan kata lain, citra pertama-tama diubah menggunakan transformasi Wavelet pada baris, dan kemudian hasilnya diubah lagi menggunakan transformasi Wavelet pada kolom. Proses ini dapat diiterasikan sesuai dengan tingkat transformasi yang diinginkan. Sementara itu, dalam pendekatan dekomposisi tidak standar, transformasi Wavelet 1 dimensi diterapkan bergantian antara kolom citra dalam langkah pertama, dan langkah ini diulang seiring dengan tingkat yang diinginkan. Pendekatan ini memberikan fleksibilitas dalam proses dekomposisi dan dapat menghasilkan representasi citra yang berbeda dari pendekatan standar.

2.3 *Convolution Neural Network*

Convolutional Neural Network (CNN) merupakan perkembangan dari metode *Multi Layer Perceptron* (MLP) [26]. CNN memiliki lebih banyak dimensi daripada MLP. CNN menerima *input* dalam bentuk array dengan dua dimensi atau lebih. Sebagai contoh, jika kita memproses gambar sepotong buah tomat, dimensi pertama mungkin menggambarkan bentuk umum tomat dan orientasinya. Sedangkan dimensi kedua mungkin merepresentasikan fitur-fitur penting seperti bagian depan dan belakang, kelopak bunga, serta tekstur permukaan tomat.

Setiap lapisan (*layer*) dalam CNN memiliki banyak fitur yang terbentuk dari lapisan sebelumnya. Fitur-fitur ini bisa mencakup informasi tentang sudut, tekstur, hingga lekukan yang signifikan. Proses konvolusi diterapkan pada setiap *filter* dengan ukuran kernel yang ditentukan. Proses *pooling* sering kali digunakan bersamaan dengan proses konvolusi. Tujuan dari proses *pooling* adalah untuk mengambil nilai perwakilan, seperti nilai maksimum, dari satu kernel.

Selain itu, metode dropout dapat diterapkan pada beberapa lapisan CNN. *Dropout* berfungsi untuk secara acak menghapus beberapa fitur yang tidak digunakan dalam proses pelatihan, membantu mencegah *overfitting*. Tahapan proses CNN dapat dilihat pada Gambar 2.2 yang menggambarkan arsitektur CNN secara umum.



Gambar 2.2 *Convolution Neural Network* (CNN)

Berdasarkan Gambar 2.2, terlihat bahwa dalam penelitian ini, model yang dikembangkan memiliki satu lapisan yang dapat menjalankan tiga metodologi secara berurutan. Hasil dari lapisan pertama disimpan dan digunakan sebagai *input* untuk lapisan kedua. Lapisan kedua kemudian dapat menjalankan metode yang sama seperti yang dilakukan oleh lapisan pertama. Jumlah lapisan yang akan digunakan dapat disesuaikan dengan kebutuhan spesifik penelitian.

Keputusan mengenai jumlah lapisan dan jumlah neuron dalam setiap lapisan adalah penting. Semakin banyak lapisan dan neuron yang ditentukan, semakin rumit modelnya, tetapi ini juga akan meningkatkan waktu komputasi yang diperlukan untuk satu iterasi metode. Oleh karena itu, penentuan jumlah lapisan dan neuron harus

mempertimbangkan *trade-off* antara kinerja model dan waktu komputasi yang tersedia. CNN mencapai tingkat akurasi yang tinggi karena memiliki sejumlah besar fitur yang diekstraksi melalui proses konvolusi, jumlah neuron, dan koneksi antar neuron yang disusun dengan bantuan bobot yang diperbarui secara berkala. Akurasi yang tinggi dapat terwujud melalui penggabungan yang optimal dari semua elemen tersebut. Kombinasi terbaik ini sering kali menjadi titik fokus utama dan dapat menghasilkan akurasi yang sangat baik. Kombinasi ini sering mengalami modifikasi dan penyesuaian untuk meningkatkan kinerja model:

a. Ukuran dari operasi konvolusi biasanya dibatasi untuk mengendalikan jumlah lapisan pada model. Semakin banyak iterasi konvolusi yang digunakan dalam jaringan, semakin lama waktu yang dibutuhkan untuk melatih dan menjalankan model tersebut. Di sisi lain, jika kita menggunakan terlalu sedikit iterasi, ini dapat memengaruhi kemampuan model untuk mendekati kebenaran atau akurasi yang diharapkan, karena jumlah fitur yang digunakan oleh model menjadi terbatas. Oleh karena itu, pemilihan ukuran dan kedalaman konvolusi harus memperhitungkan keseimbangan antara performa dan efisiensi komputasi yang diinginkan.
b. Ukuran kernel pada operasi konvolusi berfungsi sebagai sub-*matrix* yang digunakan untuk melakukan operasi konvolusi pada *input*. Pada dasarnya, ukuran kernel ini memungkinkan kita untuk merubah sejumlah nilai dalam *matrix input* menjadi satu nilai dalam *matrix output* dengan melakukan perkalian dan penjumlahan.
c. Semakin kecil ukuran kernel yang digunakan, semakin detail informasi yang dapat diambil dari *input*, dan ini dapat menghasilkan representasi yang lebih halus dan detail pada citra atau data. Namun, memang benar bahwa semakin kecil ukuran kernel, semakin banyak perhitungan yang diperlukan, yang pada gilirannya dapat membuat waktu komputasi semakin lama. Oleh karena itu, pemilihan ukuran kernel harus mempertimbangkan keseimbangan antara keakuratan representasi dan efisiensi komputasi yang diinginkan.
d. Jumlah lapisan dalam Convolutional Neural Network (CNN) berperan sebagai penampung hasil konvolusi dari *input*. Setiap lapisan dalam CNN memiliki tugas

khusus dalam mengekstraksi fitur-fitur yang semakin kompleks dari *input*. Semakin banyak lapisan yang ditambahkan dalam arsitektur CNN, semakin banyak pula fitur yang dapat dihasilkan dan dipelajari oleh model.

e. Namun, perlu diingat bahwa semakin banyak lapisan dalam jaringan juga akan meningkatkan kompleksitas model, dan ini dapat memengaruhi waktu komputasi yang diperlukan baik untuk pelatihan maupun inferensi (pengujian). Selain itu, penggunaan lapisan yang terlalu dalam juga dapat meningkatkan risiko *overfitting* jika tidak diatur dengan baik. Dalam merancang CNN, perlu dilakukan penyesuaian yang cermat untuk menemukan keseimbangan antara jumlah lapisan yang digunakan, kompleksitas model, dan waktu komputasi yang tersedia. Ini akan memastikan bahwa model memiliki performa yang baik tanpa mengorbankan efisiensi waktu komputasi. Jumlah lapisan Fully-Connected bertugas mengintegrasikan fitur-fitur yang telah diekstraksi ke dalam kelas yang sesuai. Metodenya adalah dengan memberikan nilai perkalian acak dari bobot dan bias. Ketika nilai ini masih jauh dari yang diharapkan, perkalian acak tersebut diperbarui dan diulang hingga ditemukan bobot dan bias yang optimal untuk mencapai pendekatan kelas yang tepat. Waktu yang dibutuhkan untuk proses ini cukup bervariasi, tergantung pada jumlah fitur yang dihasilkan.

f. Lapisan *Pooling* adalah komponen dalam *Convolutional Neural Network* (CNN) yang bertugas mengurangi dimensi fitur-fitur hasil ekstraksi. Melalui proses *pooling*, informasi yang paling relevan dari suatu wilayah dalam fitur-fitur tersebut diambil, sementara dimensi yang kurang relevan dihilangkan. Dalam *pooling*, biasanya digunakan metode seperti pengambilan nilai maksimum atau perhitungan nilai rata-rata untuk menggambarkan wilayah tersebut [26]

Dari penjelasan sebelumnya, terdapat banyak parameter dalam Convolutional Neural Network (CNN) yang dapat disesuaikan agar mencapai tingkat presisi dan akurasi yang tinggi, sambil juga meminimalkan waktu yang dibutuhkan. Ini dapat dicapai dengan mencari kombinasi parameter yang optimal. Hasil dari pengoptimalan parameter pada suatu dataset tertentu dapat diaplikasikan pada dataset lain, yang sering disebut sebagai *transfer learning*. Gabungan dari arsitektur *transfer learning* yang

sangat terkenal adalah Googlenet dan Alexnet. Meskipun keduanya memiliki konfigurasi model yang berbeda-beda, keduanya mampu mencapai akurasi yang sangat baik. Arsitektur Googlenet dan Alexnet telah menunjukkan performa yang sangat tinggi dan telah diuji coba pada jutaan gambar. Meskipun ada perbedaan dalam parameter *layer* dan *filter* di kedua arsitektur ini, konsep dasarnya yang digunakan adalah sama, yaitu *Convolutional Neural Network* (CNN).

2.4 YOLO

Saat ini, deteksi objek melibatkan penggunaan classifier untuk mengidentifikasi objek dalam citra dan mengevaluasinya pada berbagai lokasi dan skala. Sistem *Deformable Parts Model* (DPM) mengadopsi metode jendela geser di mana *classifier* dijalankan secara merata di seluruh citra. Di sisi lain, *Regional Convolution Neural Network* (R-CNN) menggunakan pendekatan proposal wilayah untuk pertama-tama menghasilkan kotak pembatas yang berpotensi di dalam citra dan kemudian menjalankan *classifier* pada proposal-proposal tersebut. Setelahnya, proses klasifikasi melibatkan *langkah post-processing* yang digunakan untuk menyaring kotak pembatas, menghilangkan deteksi objek yang berlebihan, dan membandingkan kotak prediksi dengan objek lainnya. Proses yang kompleks ini memerlukan waktu yang lama dan sulit untuk dioptimalkan karena setiap komponen harus dilatih secara terpisah.

YOLO mengubah pendeteksian objek menjadi sebuah permasalahan regresi tunggal, yang mengolah informasi secara langsung dari piksel gambar hingga mendapatkan koordinat kotak pembatas dan probabilitas kelasnya. YOLO dikenal sebagai sistem "*you only look once*" yang berarti bahwa sistem hanya perlu melihat gambar sekali untuk memprediksi objek apa yang ada di dalamnya dan di mana objek tersebut berada.

2.5 Evaluasi

Evaluasi performa suatu sistem klasifikasi adalah aspek yang krusial. Performa sistem klasifikasi mencerminkan sejauh mana kemampuan sistem dalam mengelompokkan data. Confusion matrix adalah salah satu alat yang digunakan untuk

mengukur performa metode klasifikasi tertentu. [28]. Pengukuran kinerja *Confussion matrix* memiliki 4 istilah representasi hasil proses klasifikasi yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) seperti pada tabel 2.1.

Tabel 2.1 Definisi Parameter TP, FP, FN, TN

	Prediction Label	
True label	TP	FN
	FP	TN

Berdasarkan Tabel 2.1 dijelaskan tentang definisi dari parameter TP, FP, FN, TN yaitu parameter yang ada dalam pengujian *confussion matrix*. TP adalah data dengan kelas positif yang terklasifikasi positif, TN adalah data dengan kelas negatif yang terklasifikasi negatif, FP adalah data dengan kelas negatif yang terklasifikasi positif dan FN adalah data dengan kelas negatif yang terklasifikasi negatif. Hasil *output* dari *confusion matrix* adalah *accuracy*, *precision*, *recall*, dan *F1score* dengan persamaan sebagai berikut.

Accuracy adalah seberapa akurat metode klasifikasi dalam menentukan kelas atau label pada data. Persamaan matematis untuk *accuracy* dapat dilihat pada Persamaan (2.1)

$$Accuracy = \frac{(TP)+(TN)}{total\ citra} X100\% \quad (2.1)$$

Persamaan (2.1) merupakan nilai *true positive* atau dapat didefinisikan sebagai data dengan kelas positif yang terklasifikasi positif. Sedangkan (TN) merupakan nilai *true negatif* atau bisa didefinisikan sebagai data dengan kelas negatif yang terklasifikasi negatif. *Accuracy* merupakan hasil bagi antara penjumlahan TP dan TN dengan jumlah kesuluruhan data citra kemudian hasil tersebut dikali 100% untuk memperoleh nilai *accuracy* dalam persen.

Precision adalah nilai fraksi tiap kelas yang diklasifikasi dengan benar. Persamaan matematis untuk *precision* dapat dilihat pada Persamaan (2.2)

$$Precision = \frac{(TP)}{(TP)+(FP)} x100\% \quad (2.2)$$

Persamaan (2.2) merupakan nilai *true* positif atau dapat didefinisikan sebagai data dengan kelas positif yang terklasifikasi positif atau hasil bagi antara TP dengan hasil penjumlahan TP dan FP kemudian hasil tersebut dikali 100% untuk memperoleh nilai *precision* dalam persen. Persamaan matematis lain untuk menghitung *precision* seperti pada Persamaan (2.3)

$$Precision = \frac{(TN)}{(TN)+(FN)} x100\% \quad (2.3)$$

Persamaan (2.3) merupakan nilai *true* negatif atau dapat didefinisikan sebagai data dengan kelas negatif yang terklasifikasi negatif. *Precision* merupakan hasil bagi antara TN dengan hasil penjumlahan TN dan FN kemudian hasil tersebut dikali 100% untuk memperoleh nilai *precision* dalam persen.

Recall adalah fraksi jumlah kelas atau label yang berhasil didapatkan dibagi dengan jumlah kelas yang seharusnya didapatkan. Persamaan matematis untuk *recall* dapat dilihat pada Persamaan (2.4)

$$Recall = \frac{(T+)}{(T+)+(F-)} x100\% \quad (2.4)$$

Persamaan (2.4) merupakan nilai *true* positif atau dapat didefinisikan sebagai data dengan kelas positif yang terklasifikasi positif. Sedangkan *recall* merupakan nilai *true* negatif atau bisa didefinisikan sebagai data dengan kelas positif yang terklasifikasi negatif. *Recall* merupakan hasil bagi antara TP dengan hasil penjumlahan TP dan FN kemudian hasil tersebut dikali 100% untuk memperoleh nilai *recall* dalam persen. Persamaan matematis lain untuk menghitung *precision* seperti pada Persamaan (2.5)

$$Recall = \frac{(TN)}{(TN)+(FP)} x100\% \quad (2.5)$$

Persamaan (2.5) TN merupakan nilai *true* negatif atau dapat didefinisikan sebagai data dengan kelas negatif yang terklasifikasi negatif. *Recall* merupakan hasil bagi antara TN dengan hasil penjumlahan TN dan FP kemudian hasil tersebut dikali 100% untuk memperoleh nilai *recall* dalam persen.

F1score adalah fungsi *harmonic mean* dari *precision* dan *recall.* Persamaan matematis untuk *f1score* dapat dilihat pada Persamaan (2.6)

$$F1score = 2\frac{Precision\ Recall}{Precision+Re}x100\% \quad (2.6)$$

Berdasarkan Persamaan (2.6) diketahu jika *f1score* merupakan 2 kali hasil bagi antara perkalian *precision recall* dengan penjumlahan antara *precision recall* yang kemudian hasil tersebut dikali 100% untuk mendapatkan nilai dalam persen.

2.6 Kajian Pustaka

Kajian pustaka membahas mengenai rangkuman dari beberapa penelitian sebelumnya yang digunakan sebagai referensi dan acuan untuk penelitian ini. Beberapa penelitian tersebut membahas mengenai metode pengolahan citra untuk deteksi retak pada jalan raya. Berikut penjabaran dari beberapa penelitian tersebut:

1. Penelitian sebelumnya memanfaatkan konsep penginderaan jauh dengan teknologi jaringan syaraf tiruan deep learning. YOLO digunakan untuk melakukan deteksi kerusakan jalan, dan hasil pendeteksian ini kemudian dikombinasikan dengan informasi posisi atau lokasi yang diperoleh melalui GNSS. Dengan demikian, hasil deteksi dapat memberikan informasi yang akurat tentang posisi atau lokasi kerusakan. Hasil penelitian ini menciptakan sebuah model identifikasi kerusakan jalan yang memiliki tingkat akurasi keseluruhan sebesar 88% dan tingkat akurasi kappa sebesar 86%, sementara untuk akurasi posisi sebaran kerusakan, koordinat posisinya memiliki akurasi RMSE sekitar ± 5,6 meter. [28].
2. Penelitian selanjutnya yang membahas mengenai deteksi retak jalan deteksi retak perkerasan dan mengusulkan metode baru untuk mendeteksi retak perkerasan menggunakan *deep learning* dengan *transfer learning*. Penelitian ini menganalisis kinerja model yang diusulkan untuk arsitektur jaringan yang berbeda yaitu, *googlenet, alexnet* dan *resent* dan menyimpulkan bahwa Googlenet memberikan kinerja yang lebih baik dalam dalam mendeteksi keretakan jalan raya [29].

3. Penelitian berikutnya tentang deteksi retak jalan menggunakan metode *localized thresholding technique* untuk melakukan klasifikasi. Metode ini tidak terbatas pada jenis tekstur perkerasan, dan metode ini efisien karena membutuhkan waktu kurang dari 20 detik per gambar untuk menghasilkan hasil. Metode ini diuji pada 130 gambar permukaan beton semen *portland* dan beton aspal. Hasil pengujian memperoleh hasil evalusi yaitu *presisi* 0,89 sedangkan *recall* 0,83, dan *F1score* 0,86 serta kurasi pengukuran panjang retak 80% [30].

4. Penelitian selanjutnya yang menggunakan metode *multi-scale retinex fused* dengan Wavelet *transform*. Algoritma *multi-scale retinex fused* dapat menutupi kekurangan dari transformasi Wavelet, sehingga kombinasi keduanya dapat memperoleh efek peningkatan retakan yang lebih baik. Selain itu, *preprocessing* berupa penghilangan bayangan dilakukan sebelum peningkatan retakan, yang secara efektif menghilangkan gangguan bayangan intensitas tinggi. Melalui perbandingan kinerja objektif, metode ini dapat menyoroti informasi retakan dengan lebih baik. Metode yang diusulkan secara efektif mewujudkan fungsi penghilangan bayangan dan peningkatan retak, sehingga pengenalan akurasi sistem deteksi secara keseluruhan mencapai 95,8%, menunjukkan bahwa algoritma tersebut memiliki signifikansi penelitian yang tinggi [31].

5. Penelitian berikutnya bertujuan untuk menggunakan CNN untuk mendeteksi mendeteksi keretakan jalan. Data dalam bentuk gambar telah digunakan sebagai *input, preprocessing*, dan segmentasi ambang batas diterapkan pada data masukan. *Output* yang diproses menggunakan CNN untuk ekstraksi fitur dan klasifikasi. Akurasi pelatihan sebesar 96,20%, akurasi validasi menjadi 96,50%, dan pengujian akurasi menjadi 94,5% [32].

Berdasarkan penjabaran dari enam penelitian yang telah dijabarkan, dapat diperoleh masukan untuk penelitian yang akan dilakukan yaitu merancang sistem untuk mendeteksi retak pada jalan raya menggunakan metode pengolahan citra. Metode pengolahan citra yag digunakan adalah Wavelet *transform* untuk ekstraksi ciri dan *Convolution Neural Network* untuk ekstraksi ciri dan klasifikasi.

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Data yang digunakan pada penelitian ini berupa gambar kondisi jalan raya dengan kondisi retak garis, retak buaya dan tidak retak. Citra diambil menggunakan kamera belakang *Handphone* Xiaomi Redmi 5 Plus dengan resolusi kamera sebesar 12MP. Pengambil data penelitian di sepanjang jalan raya Cilegon-Serang. Tahapan yang telah dilakukan sebagai berikut:

1. Melakukan Studi Literatur mengenai tema penelitian yang akan diambil.
2. Melakukan pengambilan data citra kondisi jalan sekaligus melakukan klasifikasi secara manual untuk menjadi pembanding hasil dari sistem.
3. Merancang sistem deteksi retak jalan raya menggunakan bahasa pemprograman Python.
4. Melakukan pengujian sistem yang telah dibuat terhadap citra uji untuk mendapatkan hasil akurasi progam.
5. Melakukan analisis dan pembahasan terhadap hasil pengujian sistem.
6. Menulis laporan penelitian.

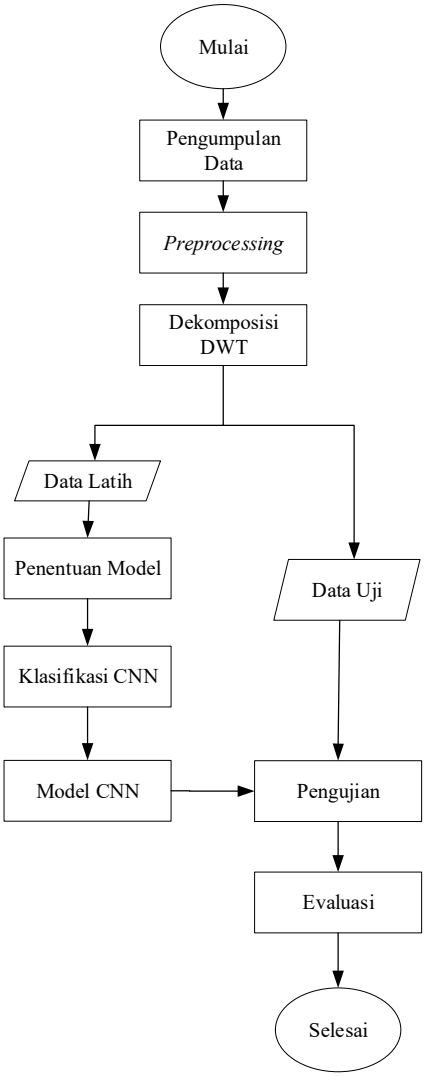
3.2 Instrumen Penelitian

Perangkat lunak dan fasilitas penunjang yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Google *Colaboratory* (GPU, RAM 12.72GB, DISK 68,40GB).
2. Google *Colaboratory*, untuk membuat *virtual environments* dan memasang paket yang diperlukan untuk proses *deep learning*.
3. Python versi 3.7 sebagai bahasa pemograman.
4. PyCharm *Community* untuk pembuatan GUI.
5. Laptop ASUS X455L

3.3 Perancangan Penelitian

Flowchart penelitian memiliki peran penting dalam menyajikan informasi mengenai seluruh proses penelitian, mulai dari tahap awal hingga penyelesaian. Pada Gambar 3.1, terdapat ringkasan visual dari tahapan-tahapan yang dilakukan selama penelitian. Ini adalah representasi visual yang memberikan gambaran umum tentang penelitian yang telah diselesaikan..

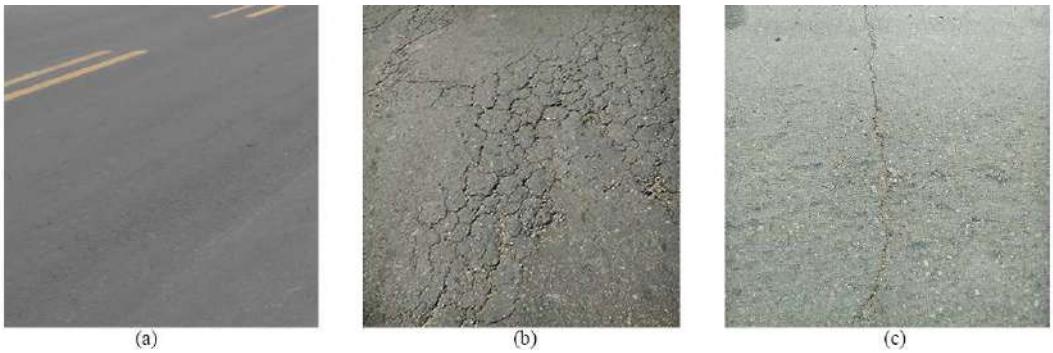


Gambar 3.1 *Flowchart* Penelitian

Berdasarkan Gambar 3.1 dijelaskan tahapan alur penelitian untuk Implementasi metode Wavelet dan *backpropagation* pada deteksi retak jalan raya berbasis pengolahan citra *neural network backpropagation* yaitu, pengumpulan data, *preprocessing*, dekomposisi DWT, pelatihan, pengujian dan evaluasi.

3.4 Pengumpulan Data

Pada tahap ini, kami menggunakan 219 citra hasil foto retak jalan raya aspal dengan format (.jpg) sebagai citra masukan. Kumpulan data ini terdiri dari 73 data retak buaya, 73 data retak garis, dan 73 data yang tidak mengalami retakan. Citra-citra hasil foto kondisi retak jalan raya ini kemudian diproses menggunakan metode pengolahan citra, dan hasilnya dapat dilihat dalam Gambar 3.2.



Gambar 3.2 Kondisi Jalan (a) Tidak Retak (b) Retak Buaya (c) Retak Garis

Gambar 3.2 merupakan citra dari kondisi jalan tidak retak, (b) kondisi jalan mengalami retak buaya dan, (c) kondisi jalan mengalami retak garis. Setiap dari citra tersebut memiliki ciri khusus yang diproses ekstraksi ciri.

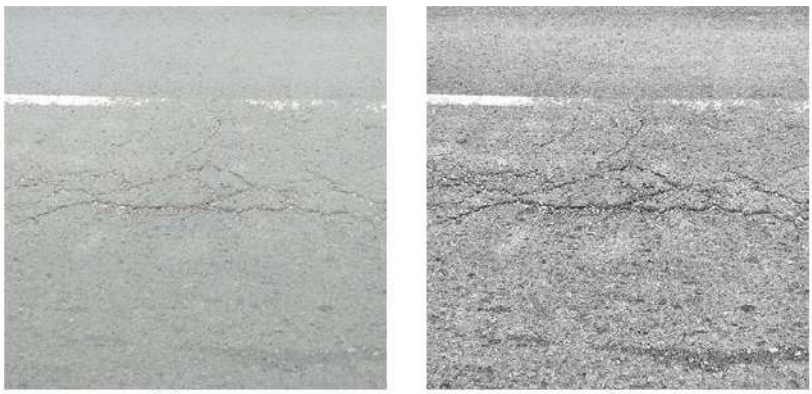
3.5 Preprocessing

Preprocessing merupakan langkah awal dalam memproses citra sebelum melanjutkan ke tahapan Dekomposisi DWT dan *Neural Network*. Terdapat tiga tahap dalam proses preprocessing ini, yakni *resizing, kontrast adjustment, dan grayscaling.*

Proses *resizing* digunakan untuk menyamakan ukuran semua citra menjadi berukuran 512x512 piksel. Ini penting karena citra asli dapat bervariasi dalam ukuran.

Selanjutnya, data citra sering dipengaruhi oleh faktor seperti intensitas pencahayaan dan pengaturan kamera. Untuk mengatasi ini, dilakukan penyesuaian kontras menggunakan metode *contrast limited adaptive histogram equalization* (CLAHE) untuk meningkatkan kontras citra tanpa mengorbankan kualitasnya.

Proses *grayscaling* dimulai dengan mengambil citra RGB sebagai *input*. Dari citra RGB ini, tinggi dan lebar citra diidentifikasi untuk menciptakan citra baru. Kemudian, nilai komponen *red*, *green* dan *blue* dipisahkan untuk membentuk citra baru yang menggambarkan perubahan dalam model warna, seperti yang ditunjukkan dalam Gambar 3.3.



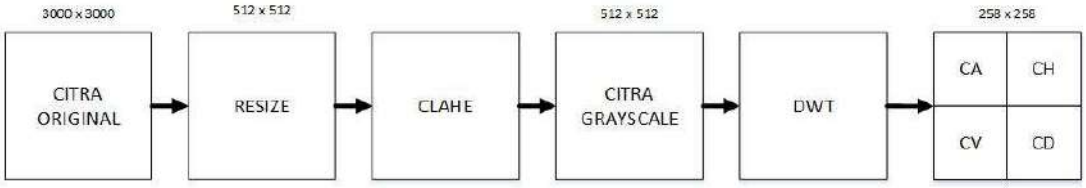
Gambar 3.3 Jenis Citra (a) RGB (b) *Grayscale*

Citra yang terdapat dalam Gambar 3.3 adalah contoh citra hasil *grayscaling* yang akan berfungsi sebagai data *input* pada tahap ekstraksi ciri menggunakan proses dekomposisi DWT.

3.6 Ekstraksi Ciri Menggunakan Dekomposisi DWT

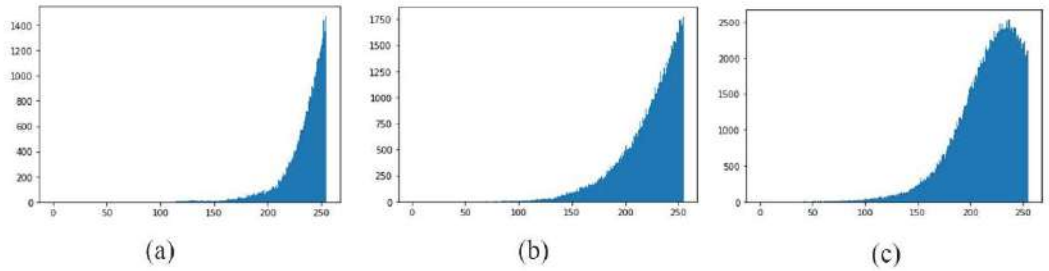
Dekomposisi *discrete Wavelet* transform adalah proses mengurai citra menjadi subgambar (subband) yang memiliki frekuensi dan orientasi berbeda, yang terdiri dari

low-low (LL), *low-high* (LH), *high-low* (HL), dan *high-high* (HH). Penerapan *discrete Wavelet transform* dilakukan dengan menggunakan sinyal *filter* frekuensi tinggi (*highpass filter*) dan frekuensi rendah (*lowpass filter*). Langkah-langkah dalam proses dekomposisi *Wavelet* dapat ditemukan dalam Gambar 3.4 sebagai ilustrasi.



Gambar 3.4 Proses Ekstraksi Ciri

Dari ilustrasi dalam Gambar 3.4, kita dapat mengidentifikasi langkah-langkah dalam proses ekstraksi ciri sebagai berikut: Awalnya, citra asli atau citra RGB digunakan sebagai *input*. Tahap berikutnya adalah melakukan *resizing* untuk menghasilkan citra dengan ukuran seragam, yaitu 512 x 512 piksel, dan kemudian meningkatkan kontras citra menggunakan metode CLAHE. Setelah kontras diperbaiki, citra RGB diubah menjadi citra *grayscale*. Citra *grayscale* ini digunakan sebagai *input* dalam proses *transformasi Wavelet,* yang menghasilkan empat jenis koefisien, yaitu *Coefficient Approximation* (CA), *Coefficient Horizontal* (CH), *Coefficient Vertical* (CV), dan *Coefficient Diagonal Detail* (CD). Hasil dari transformasi ini digunakan sebagai *input* dalam proses CNN. Penting untuk dicatat bahwa setiap citra *input* memiliki histogram nilai yang berbeda, seperti yang ditunjukkan dalam Gambar 3.5.



Gambar 3.5 menunjukkan histogram citra untuk tiga jenis kondisi: tidak retak (a), retak garis (b), dan retak buaya (c). Dari Gambar 3.5, dapat disimpulkan bahwa citra-citra yang menggambarkan kondisi jalan yang berbeda, seperti tidak retak, retak garis, dan

retak buaya, memiliki karakteristik histogram yang berbeda-beda. Hal ini mengindikasikan bahwa setiap citra memiliki ciri khusus yang dapat digunakan untuk mengklasifikasikan jenis kondisinya berdasarkan hasil ekstraksi cirinya.

3.7 Pembagian Data

Langkah berikutnya melibatkan pembagian dataset yang telah ada menjadi dua bagian: data pelatihan dan data pengujian menggunakan model yang sudah tersedia dalam pustaka *Python* bernama scikit-learn (*sklearn*). Penelitian ini memilih menggunakan pustaka *sklearn* karena memiliki kode yang lebih ringkas dibandingkan dengan pustaka lain seperti *tensorflow* atau *numpy*. Pustaka *sklearn* juga sering digunakan untuk melakukan pengelompokan data dengan karakteristik serupa ke dalam kelompok yang sama dan data dengan karakteristik yang berbeda ke kelompok yang berbeda.

Untuk melakukan pembagian ini, program menggunakan fungsi *train_test_split* yang memisahkan array atau *matrix* menjadi data pelatihan dan data pengujian secara acak. Proses ini menetapkan test_size=0.2, yang berarti 80% dari dataset digunakan untuk data pelatihan dan 20% untuk data validasi. Pembagian dataset dilakukan sesuai dengan jenis datanya, sebagaimana yang tercantum dalam Tabel 3.1.

Tabel 3.1 Jumlah *Dataset* Hasil DWT

Jenis data	jumlah	Keterangan
Training	(560, 258x258, 1)	Jumlah,*size, channel*
Validation	(140, 258x258, 1)	Jumlah,*size, channel*
Testing	(176, 258x258, 1)	Jumlah,*size, channel*

Berdasarkan Tabel 3.1 *dataset* dibagi menjadi tiga kelompok, yaitu data *training*, data *validation*, dan data *testing*.

3.8 Pembangunan Arsitektur Model *Neural Network*

Pengembangan struktur jaringan CNN memiliki potensi untuk memengaruhi tingkat akurasi yang dicapai oleh model. Seperti yang terlihat pada (Lampiran A Gambar A-1), arsitektur CNN merupakan kerangka kerja jaringan yang digunakan

dalam tahap pelatihan dengan tujuan mencapai model yang optimal. Dalam penelitian ini, citra *input* yang digunakan memiliki dimensi 258x258x1. Deskripsi lebih lanjut mengenai arsitektur yang ada pada Lampiran A-1 akan dijelaskan di bawah ini.:

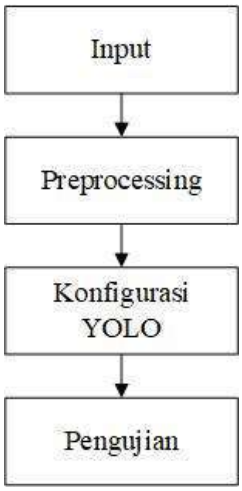
1. Pada langkah pertama *konvolusi*, digunakan kernel berukuran 3x3 dengan total 32 *filter*. *Konvolusi* adalah proses kombinasi antara dua *matrix* yang berbeda untuk menghasilkan *matrix* nilai baru. Setelah tahap *konvolusi* selesai, langkah selanjutnya adalah menerapkan fungsi aktivasi ReLU. Fungsi aktivasi ini berfungsi untuk mengubah nilai-nilai negatif menjadi nol atau menghapus nilai-nilai negatif dalam *matrix* hasil konvolusi tersebut. Ukuran *matrix* hasil konvolusi tetap sama, yaitu 258x258, karena digunakan *padding* dengan nilai 0 selama proses *konvolusi*.
2. *Pooling* merupakan tahap di mana ukuran *matrix* dikurangi melalui operasi *pooling*. Proses *pooling* ini melibatkan penggunaan sebuah *filter* dengan ukuran tertentu yang bergerak secara bergantian pada seluruh area *feature map*. Dalam penelitian ini, metode yang digunakan adalah *max-pooling* untuk menghasilkan *matrix* nilai baru setelah proses *pooling*. Hasil *pooling* ini menghasilkan *matrix* baru dengan ukuran 129x129, dengan menggunakan *filter pooling* berukuran 2x2. Prinsip kerja *max-pooling* adalah dengan mengambil nilai maksimum dari setiap pergeseran kernel, yang dilakukan sebanyak nilai stride, dalam hal ini adalah 2.
3. Langkah *konvolusi* kedua adalah melanjutkan dari hasil *pooling* pertama. Pada tahap ini, digunakan *matrix* gambar berukuran 129x129 sebagai *input*, dengan 32 *filter*, dan menggunakan kernel berukuran 3x3. Seperti halnya pada *konvolusi* sebelumnya, *konvolusi* kedua ini juga menerapkan fungsi aktivasi ReLu.
4. Langkah berikutnya adalah memasuki tahap *pooling* kedua, yang memiliki kesamaan dengan *pooling* pertama, namun ada perbedaan dalam ukuran *matrix output* akhirnya. Hasil *pooling* kedua menghasilkan *matrix* dengan ukuran gambar 64x64.

5. Langkah *konvolusi* ketiga adalah melanjutkan dari hasil *pooling* pertama. Pada tahap ini, digunakan *matrix* gambar berukuran 64x64 sebagai *input*, dengan 64 *filter*, dan menggunakan kernel berukuran 3x3. Seperti sebelumnya, *konvolusi* ketiga ini juga menerapkan fungsi aktivasi ReLu.
6. Langkah berikutnya adalah memasuki tahap *pooling* ketiga, yang hampir identik dengan *pooling* pertama dan kedua, namun memiliki perbedaan dalam ukuran *matrix output* akhirnya. Hasil dari *pooling* ketiga menghasilkan *matrix* dengan ukuran gambar 32x32.
7. Langkah berikutnya adalah proses *Flatten* atau *fully connected.* Pada tahap ini, digunakan hanya satu lapisan tersembunyi dalam jaringan MLP (*Multi Layer Perceptron*). *Flatten* di sini mengubah keluaran dari lapisan *pooling* menjadi vektor tunggal. Sebelum memulai proses klasifikasi atau prediksi gambar, tahap ini melibatkan penggunaan nilai *dropout*. *Dropout* merupakan sebuah teknik pengaturan dalam jaringan saraf yang bertujuan untuk memilih beberapa neuron secara acak dan tidak menggunakannya selama proses pelatihan. Dengan kata lain, neuron-neuron tersebut dieliminasi secara acak. Tujuan dari tahap ini adalah untuk mengurangi *overfitting* selama proses pelatihan.
8. Langkah terakhir melibatkan penggunaan fungsi aktivasi *Softmax*. Fungsi ini memiliki aplikasi khusus dan umumnya digunakan dalam metode klasifikasi seperti regresi logistik multinomial dan analisis diskriminan *linier multikelas*.

3.9 Deteksi Kondisi Retak Jalan Raya Secara *Real-time*

Secara garis besar, langkah-langkah dalam penelitian ini dimulai dengan mengumpulkan data citra, kemudian melakukan praproses data citra, yang mencakup pelabelan dan *resizing* citra. Selanjutnya, konfigurasi jaringan YOLO disesuaikan dengan data citra dan dilakukan pelatihan untuk membentuk model YOLO yang baru.

Langkah terakhir dalam penelitian adalah menguji model menggunakan data video.Proses yang terlibat dalam YOLO dijelaskan dalam Gambar 3.6

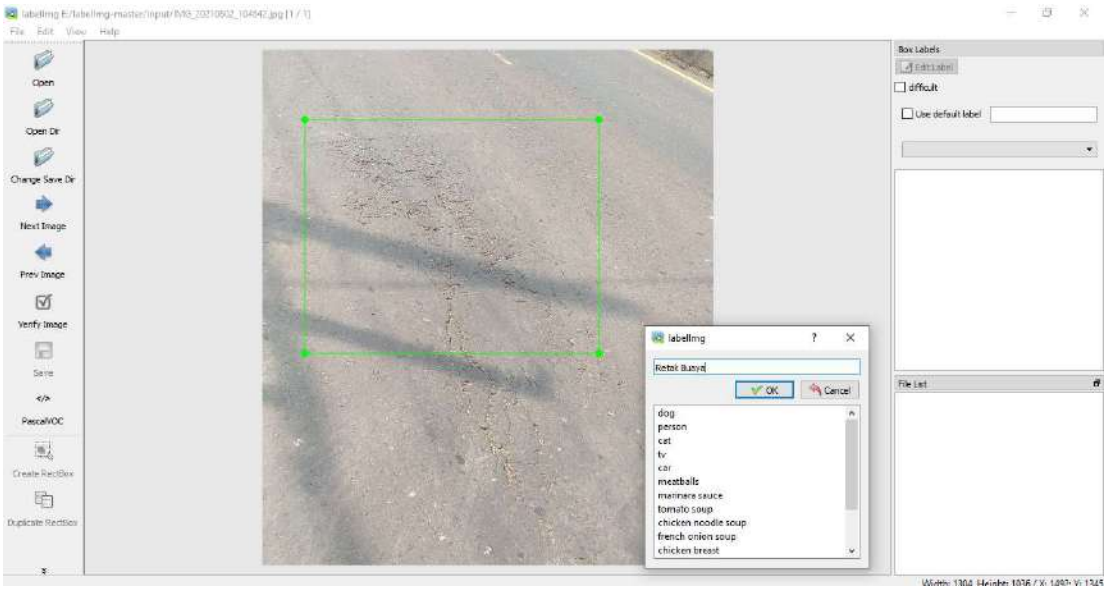


Gambar *3.5 Flowchart* Deteksi Retak Jalan Secara *Real-Time*

Dalam Gambar 3.6, tahapan proses pembuatan model YOLOv4 mencakup empat langkah: *input, preprocessing,* konfigurasi YOLO, dan pengujian. Berikut adalah penjelasan untuk masing-masing langkahnya. *Input* yang digunakan dalam deteksi retakan jalan secara *real-time* adalah video yang menampilkan kondisi jalan raya. Proses *preprocessing* data melibatkan dua aspek, yaitu pelabelan dan penyesuaian ukuran citra. Pelabelan citra merupakan langkah awal di mana setiap citra dalam dataset diberi label untuk menyimpan informasi tentang citra tersebut. Ini dilakukan dengan menambahkan *bounding box* dan memberikan nama kelas pada setiap objek dalam citra. Selanjutnya, dilakukan perubahan ukuran citra untuk meningkatkan

kinerja model YOLO dalam pengenalan objek. Proses pelabelan *bounding box* dapat dilihat dalam Gambar 3.7 sebagaimana yang dijelaskan di bawah ini

.



Gambar 3.6 Pelabelan Data

Gambar 3.7 merupakan *interface* dari proses *labeling* dari data yang dijadikan *input* dari YOLO. Seluruh data di labeli satu persatu keseluruhan tanpa terkecuali.

a. Konfigurasi Jaringan YOLO

Konfigurasi parameter yang digunakan untuk pembentukan model YOLOv4 pada Tabel 3.2 sebagai berikut.

Tabel 3.2 Konfigurasi YOLO

Parameter	**Nilai**
Batch	64
Subdivisions	16
Width	320
Height	320
Channels	3
Momentum	0,949
Decay	0,0005

Parameter	Nilai
Angle	0
Saturation	1.5
Exposure	1.5
Hue	1
learning_rate	0,001
burn_in	1000
max_batches	2210
Policy	Steps
Scales	1; 1

Tabel 3.2 merupakan pengaturan parameter untuk proses *training* data yang akan dilakukan. Setiap parameter sangat mempengaruhi hasil akhir dari model yang dibuat sehingga diperlukan penentuan parameter terbaik.

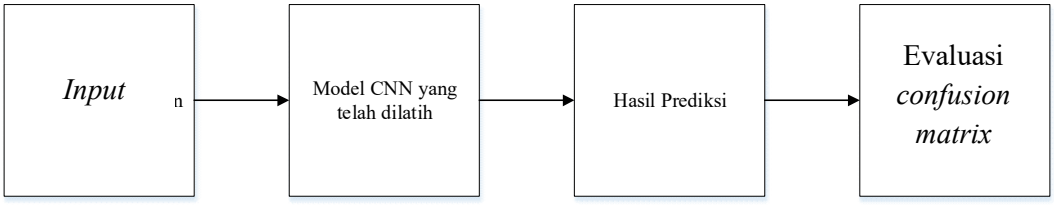
b. Pengujian Model YOLO

Proses pengujian dilakukan dengan menggunakan video sebagai *input*. Video ini berasal dari rekaman kondisi jalan raya di kota Cilegon yang mencakup kondisi jalan yang mengalami retakan. Pengujian bertujuan untuk mengevaluasi akurasi dalam mendeteksi objek menggunakan model yang telah dilatih sebelumnya.

3.10 Pelatihan dan Pengujian

Pengujian adalah suatu tahap di mana sistem atau program dijalankan dalam kondisi tertentu, dilakukan pengamatan terhadap hasil pengujian, dan dilakukan evaluasi terhadap komponen-komponen yang mungkin kurang optimal. Selain itu, pengujian juga berperan dalam mengidentifikasi dan mengungkapkan potensi kesalahan yang mungkin terjadi, serta mencari solusi untuk mengatasi kesalahan tersebut. Dalam beberapa kasus, pengujian perlu diulang untuk memastikan perbaikan telah terjadi.

Dalam konteks program deteksi retakan pada jalan raya, pengujian bertujuan untuk menghasilkan program deteksi retakan yang memiliki kinerja optimal. Alur pengujian sistem dapat dilihat dalam Gambar 3.8. Hasil dari pengujian ini dinyatakan dalam bentuk persentase tingkat akurasi, presisi, *recall*, dan skor f1 yang dihasilkan dari evaluasi menggunakan *matrix* kebingungan (confusion matrix) hasil pengujian.



Gambar 3.7 Alur Pengujian Sistem

Berdasarkan Gambar 3.8 penelitian ini dilakukan tiga pengujian yaitu:

a. Pelatihan dan Pengujian model CNN

Dalam referensi pada Lampiran A.1 mengenai arsitektur CNN, dilakukan proses pelatihan dengan tujuan memperoleh model CNN yang memiliki kinerja yang baik dan menghindari masalah *overfitting* dan *underfitting*. Setelah melalui tahap pelatihan, model yang telah dihasilkan kemudian diuji menggunakan *matrix* kebingungan (*confusion matrix*) untuk mengukur kinerja. Penilaian apakah sebuah model klasifikasi efektif atau tidak, dapat dilihat melalui parameter evaluasi kinerjanya, seperti akurasi, *recall*, presisi, dan nilai f1. Untuk menghitung parameter-parameter ini, digunakan *matrix* kebingungan yang mencakup berbagai nilai, seperti *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Matriks ini mencakup seluruh kemungkinan hasil yang benar positif (P) dan benar negatif (N).

b. Pengujian Menggunakan Variasi *Epoch*

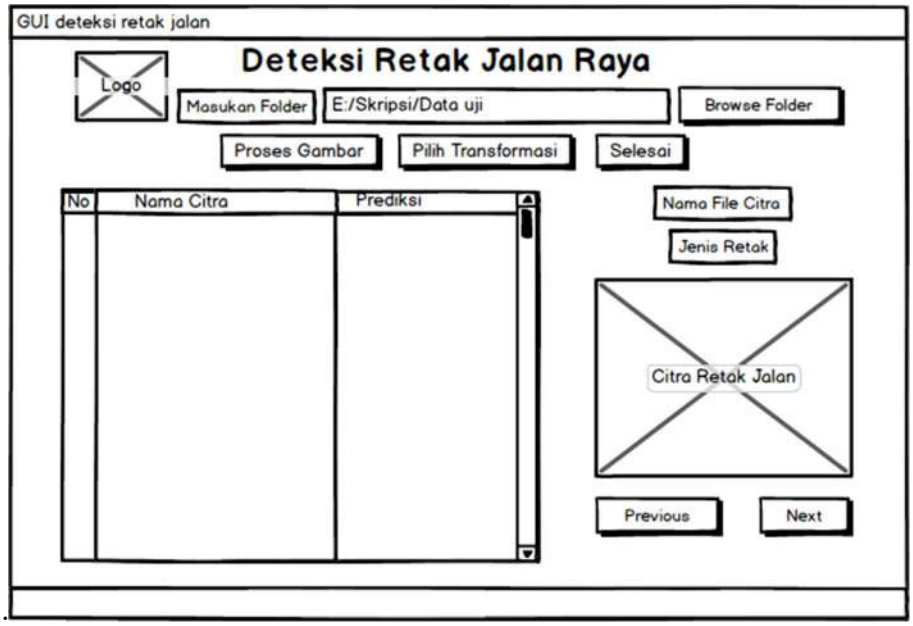
Model CNN yang telah sebelumnya dibuat, kembali menjalani serangkaian pengujian dengan mengubah parameter tertentu. Dalam pengujian ini, parameter yang dimodifikasi adalah nilai *epoch*. Terdapat tiga nilai *epoch* yang digunakan, yakni 100, 300, dan 500. Hasil dari proses pelatihan setiap model menunjukkan performa yang memuaskan karena tidak terjadi overfitting maupun underfitting terhadap data yang digunakan. Lampiran D.1 hingga D.3 menyajikan detail mengenai proses pelatihan dan grafik *loss* serta akurasi yang dihasilkan dari pelatihan dan validasi dengan variasi *epoch* 100, 300, dan 500.

c. Pengujian menggunakan variasi *learning rate*.

Dilakukan kembali pengujian model dengan *set* parameter berbeda. Pengujian ini menggunakan parameter pembeda yaitu nilai *learning rate*. Nilai *learning rate* yang digunakan yaitu 0,01, 0,001, dan 0,0001. Hasil dari *training* setiap model menunjukan performa yang baik karena tidak adanya *overfitting* dan *underfitting* pada data yang dilatih. Lampiran D.4 sampai D.6 menunjukan proses pelatihan dan grafik *loss* dan akurasi hasil *training* dan validasi pelatihan dengan variasi *learning rate*.

1. Pengujian GUI.

Graphical User Interface (GUI) adalah antarmuka visual pada sistem komputer yang menggunakan menu dan elemen grafis. Maksud dari pembuatan GUI adalah untuk menyederhanakan pengoperasian program bagi pengguna dengan menyediakan tampilan yang lebih intuitif. GUI adalah perancangan antarmuka yang kemudian diwujudkan dalam pembuatan program. Detail mengenai rancangan GUI yang telah dibuat dapat ditemukan dalam Gambar 3.9



Gambar 3. 8 GUI Deteksi Retak Jalan

Gambar 3.9 menggambarkan desain GUI yang mencakup beberapa informasi, seperti nama program dan tombol "browse folder" yang digunakan untuk memilih citra yang akan diuji, kemudian citra-citra ini ditampilkan dalam daftar. Citra asli sebelum

mengalami proses transformasi juga ditampilkan, sehingga pengguna dapat membandingkan citra hasil prediksi dengan klasifikasi aktual dari citra tersebut. Rancangan ini juga mencakup hasil keputusan program, yaitu klasifikasi citra yang telah diproses, apakah citra tersebut termasuk dalam kategori citra retak atau citra tidak retak. Selanjutnya, dilakukan pengujian untuk mengevaluasi hasil akurasi deteksi retak jalan raya menggunakan GUI ini. Parameter yang diukur dalam skenario pengujian sistem ini terdiri dari akurasi, presisi, recall, dan *skor f1* dalam mendeteksi kondisi retak.

2.Pengujian model YOLO

Dataset pengujian yang telah terkumpul kemudian dianalisis dengan menggunakan bobot yang telah dilatih sebelumnya (trained weights). Nilai *threshold* sebesar 0,4 diaplikasikan ke seluruh dataset pengujian. Proses pengujian melibatkan langkah-langkah seperti persiapan model jaringan, penentuan *threshold*, dan penggunaan bobot yang telah disiapkan. Setelah model dan bobot telah dibangkitkan dan disesuaikan dengan *pre-trained weights* yang telah dilatih menggunakan satu kelas objek, data pengujian dapat langsung dievaluasi menggunakan program yang telah dimasukkan dengan berkas bobot baru tersebut. Program tersebut kemudian dijalankan untuk mengevaluasi sejauh mana bobot yang telah dilatih sebelumnya mampu mendeteksi dan menghitung jumlah orang pada data pengujian. Dalam skenario pengujian sistem ini, parameter yang diukur melibatkan *akurasi, presisi, recall, dan skor f1*.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil *Training* Dan Validasi

Pengujian dilakukan untuk melakukan evaluasi terhadap model yang dihasilkan oleh CNN. Pengujian model dilakukan untuk model tiga kelas dalam membedakan kondisi jalan retak buaya, retak garis dan tidak retak pengujian ini dilakukan dengan variasi *epoch* sebanyak 100, 300, 500 dan pelatihan dengan variasi *learning rate* 0,01, 0,001, 0,0001.

4.1.1 Pelatihan dengan Variasi *Epoch*

Pada penelitian ini pelatihan model CNN dilakukan dengan variasi beberapa nilai *epoch*, yaitu 100, 300, dan 500. Hasil dan grafik dari pelatihan yang telah dilakukan dapat dilihat pada lampiran D dan hasil dari pelatihan tersebut disajikan pada Tabel 4.1 sebagai berikut.

Tabel 4.1 Perbandingan Jumlah Epoch Terhadap Loss *dan* Accuracy

Epoch	**100**	**300**	**500**
Training loss	2.1287	0,00	0,00
Validas*i loss*	0,2510	0,421	0,3636
Training Accuracy	100%	100	100%
Validasi *Accuracy*	95%	93.57%	96.43%

Tabel 4.1 merupakan perbandingan nilai *loss*, akurasi, validasi *loss*, dan validasi akurasi dengan beberapa jumlah *epoch* yaitu 100, 300, dan 500, Berdasarkan Berdasarkan Tabel 4.1, *training loss* terkecil terjadi pada *epoch* 300 dan 500 yaitu 0,00 *dan training accuracy* pada setiap jumlah *epoch* yang diuji bernilai sama yaitu 100%, sedangkan validasi akurasi tertinggi terjadi pada saat *epoch* 500, validasi *loss* terendah terjadi pada saat *epoch* 100, dan validasi akurasi tertinggi terjadi pada 500 *epoch*. Pelatihan yang sudah dilakukan menyatakan bahwa pada 500 *epoch* memperoleh akurasi yang optimum. Berdasarkan data di atas jumlah Jumlah *epoch* dapat

mempengaruhi nilai akurasi. Hal ini terlihat pada tabel 4.1 diketahui bahwa semakin banyak jumlah *epoch* yang diberikan maka akan meningkatkan akurasi. Dalam beberapa kasus, terkadang semakin besar jumlah *epoch* tidak membuat akurasi naik, hal ini terjadi pada jumlah *epoch* 300, dimana nilai akurasinya malah turun. Terdapat beberapa parameter yang mempengaruhi kondisi tersebut, diantranya jumlah *filter* sebanyak 32 pada lapisan konvolusi pertama dan kedua dengan ukuran *kernel* pada lapisan pertama dan kedua sebesar 3x3.

4.1.2 Pelatihan dengan Variasi *Learning Rate*

Pelatihan model CNN dilakukan dengan variasi beberapa nilai *learning rate*, yaitu 0,01, 0,001, dan 0,0001. Hasil dan grafik dari pelatihan yang telah dilakukan dapat dilihat pada lampiran D. Hasil dari pelatihan tersebut disajikan pada Tabel 4.2 sebagai berikut.

Tabel 4.2 Perbandingan Nilai *Learning Rate* Terhadap *Loss* Dan *Accuracy*

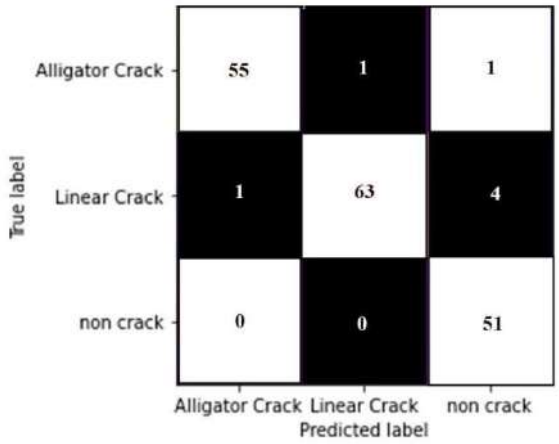
Learning rate	**0,01**	**0,001**	**0,0001**
Training loss	0,00	0,00	0,00
Training Accuracy	100%	100%	100%
Validasi loss	0,6413	0,3636	0,26
Validasi Accuracy	93.57%	96.43%	95%

Berdasarkan tabel 4.2 *learning rate* sangat berpengaruh pada performa akurasi. penggunaan nilai *learning rate* 0,01 menghasilkan nilai akurasi yang paling rendah yaitu sebesar 93,57% dan *loss* paling besar yaitu 0,6413 sehingga semakin besar *learning rate*, maka menghasilkan nilai *loss* yang besar ketika menjalankan iterasi pada saat pelatihan. Ketika menggunakan nilai *learning rate* 0,001 tingkat akurasi yang dihasilkan besar yaitu 96,43% dan nilai *loss* nya semakin sedikit yaitu 0,26. Penyebab dari hal ini karena adanya beberapa nilai *loss* yang mulai menurun dalam beberapa iterasi. Berbeda dengan penggunaan *learning rate* 0,0001 yang menunjukan tingkat akurasi yang menurun yaitu sebesar 95% disebabkan oleh lambatnya proses konvergensi nilai *loss* pada saat proses *training* serta dari segi waktu, semakin kecil

learning rate maka semakin lama waktu yang digunakan untuk *training.*

4.2 Hasil Pengujian Model CNN

Pengujian model CNN dilakukan untuk mengetahui kemampuan model dalam mengklasifikasi gambar. Evaluasi ini didapat dari hasil klasifikasi pada data validasi untuk memudahkan dalam mengevaluasi model dalam mengklasifikasi data validasi maka ditampilkan hasil klasifikasi dalam *confusion matrix*. *Confusion matrix* dapat memperlihatkan berapa banyak data yang terklasifikasi dengan benar dan terklasifikasi tidak benar. Hasil dari pengujian model CNN ditunjukan oleh *confusion matrix* seperti pada Gambar 4.1.



Gambar 4.1 *Confusion Matrix* Hasil Pengujian

Berdasarkan Gambar 4.1 dapat dilihat bahwa data validasi tidak seluruhnya mampu terklasifikasi dengan benar sesuai labelnya. Data berlabel retak buaya ada 55 data yang terklasifikasi dengan benar, 1 data terklasifikasi retak garis, dan 1 data terklasifikasi tidak retak sedangkan data yang berlabel retak garis ada 63 data terklasifikasi dengan benar, 1 data terklasifikasi sebagai kelas retak buaya, dan 4 data terklasifikasi pada kelas tidak retak. Data yang berlabel kelas tidak retak ada 51 data yang terklasifikasi dengan benar, 0 data terklasifikasi pada kelas retak buaya, dan 0 data terklasifikasi pada kelas retak garis. Hasil evaluasi performa dari pengujian disajikan pada Gambar 4.2 sebagai berikut.

	precision	recall	f1-score	support
Alligator Crack	0.98	0.96	0.97	57
Linear Crack	0.98	0.93	0.95	68
non crack	0.91	1.00	0.95	51
accuracy			0.96	176
macro avg	0.96	0.96	0.96	176
weighted avg	0.96	0.96	0.96	176

Gambar 4.2 Evaluasi Model CNN

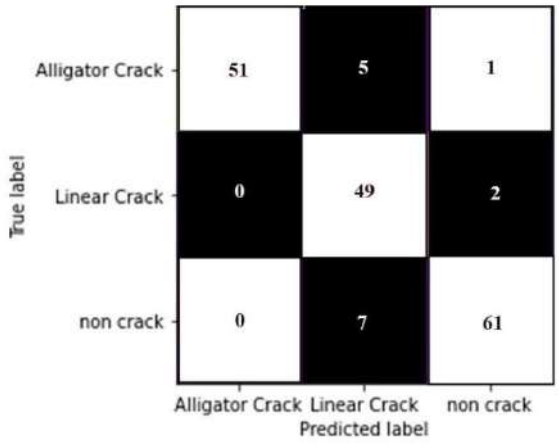
Berdasarkan Gambar 4.2 hasil dari nilai prediksi pada data validasi yang benar menghasilkan nilai yang cukup besar dibandingkan dengan prediksi data validasi yang salah. Perbandingan jumlah data yang terprediksi benar dan salah, tingkat keberhasilan sistem memprediksi data dengan benar mendapatkan akurasi sebesar 0,96 seperti pada Gambar 4.2. Evaluasi kinerja model dalam mengklasifikasi data validasi mendapatkan nilai *precision* sebesar 0,96 seperti pada Gambar 4.2, dan nilai *recall* sebesar 0,96. Perlu dilakukan perhitungan *f1score* untuk dipertimbangkan menjadi acuan dalam tingkat keberhasilan model dalam mengklasifikasi jenis retak jalan.

Nilai yang didapat dari perhitungan *f1score* di atas adalah 0,96. Nilai *f1score* dapat dipertimbangkan menjadi acuan performa model klasifikasi apabila *false negative* dan *false positive* data klasifikasi nilanya tidak saling mendekati. Nilai *f1score* adalah 0,96 dan nilai akurasi adalah 0,96 sehingga keduanya sama dan dapat menjadi acuan performa klasifikasi retak jalan raya.

Nilai yang didapatkan dari klasifikasi ini sudah mendekati angka 100%. Penelitian ini dilakukan semaksimal mungkin untuk mencapai nilai yang paling optimal

4.2.1 Pengujian model CNN dengan pelatihan 100 *epochs*

Hasil dari pengujian model CNN dengan parameter pelatihan 100 *epochs* disajikan pada *confusion matrix* seperti pada Gambar 4.3 sebagai berikut.



Gambar 4.3 *Confusion Matrix* Hasil Pelatihan dengan 100 *Epochs*

Berdasarkan Gambar 4.3 dapat dilihat bahwa data validasi tidak seluruhnya mampu terklasifikasi dengan benar sesuai labelnya. Data berlabel retak buaya ada 51 data yang terklasifikasi dengan benar, 5 data terklasifikasi retak garis, dan 1 data terklasifikasi tidak retak. Data yang berlabel retak garis ada 49 data terklasifikasi dengan benar, 0 data terklasifikasi sebagai kelas retak buaya, dan 2 data terklasifikasi pada kelas tidak retak. Data yang berlabel kelas tidak retak ada 61 data yang terklasifikasi dengan benar, 0 data terklasifikasi pada kelas retak buaya, dan 7 data terklasifikasi pada kelas retak garis. Gambar 4.2 merupakan hasil evaluasi performa pada pengujian 100 *epochs.*

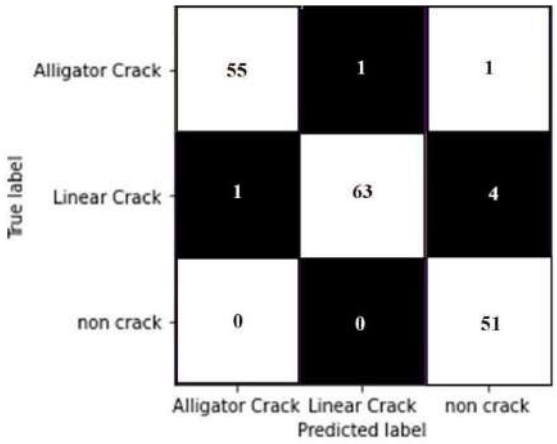
	precision	recall	f1-score	support
retak buaya	1.00	0.89	0.94	57
retak garis	0.80	0.96	0.88	51
tidak retak	0.95	0.90	0.92	68
accuracy			0.91	176
macro avg	0.92	0.92	0.91	176
weighted avg	0.92	0.91	0.92	176

Gambar 4.4 Evaluasi Model CNN dengan Pelatihan 100 *Epochs*

Gambar 4.4 merupakan hasil evaluasi performa *confusion matrix* hasil dari pengujian diperoleh *accuracy* 0.91 *precision* 0,92, *recall* 0,92, dan *f1score* 0,91.

4.2.2 Pengujian model CNN dengan pelatihan 300 *epochs*

Hasil dari pengujian model CNN dengan parameter pelatihan 300 *epochs* disajikan pada *confusion matrix* seperti pada Gambar 4.5 sebagai berikut.



Gambar 4.5 Confusion Matrix Hasil Pelatihan dengan 300 *Epochs*

Berdasarkan Gambar 4.5 dapat dilihat bahwa data validasi tidak seluruhnya mampu terklasifikasi dengan benar sesuai labelnya. Data berlabel retak buaya ada 55 data yang terklasifikasi dengan benar, 1 data terklasifikasi retak garis, dan 1 data terklasifikasi tidak retak. Data yang berlabel retak garis ada 63 data terklasifikasi dengan benar, 1 data terklasifikasi sebagai kelas retak buaya, dan 4 data terklasifikasi pada kelas tidak retak. Data yang berlabel kelas tidak retak ada 51 data yang terklasifikasi dengan benar, 0 data terklasifikasi pada kelas retak buaya, dan 0 data terklasifikasi pada kelas retak garis. Hasil evaluasi performa *confusion matrix* ditunjukan pada gambar 4.6.

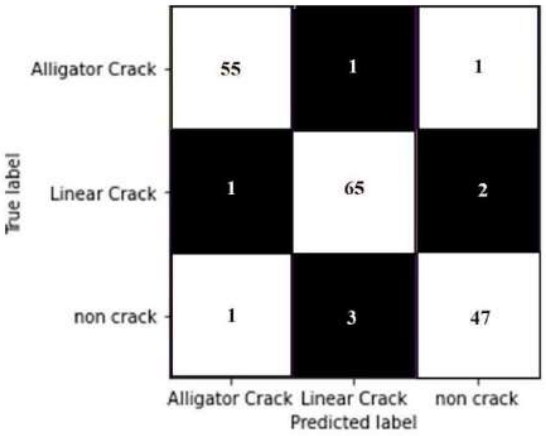
	precision	recall	f1-score	support
Alligator Crack	0.98	0.96	0.97	57
Linear Crack	0.98	0.93	0.95	68
non crack	0.91	1.00	0.95	51
accuracy			0.96	176
macro avg	0.96	0.96	0.96	176
weighted avg	0.96	0.96	0.96	176

Gambar 4.6 Hasil Evaluasi Model CNN dengan Pelatihan *300 Epochs*

Gambar 4.6 merupakan hasil evaluasi performa *confusion matrix* hasil dari pengujian diperoleh *accuracy* 0,96, *precision* 0,96, *recall* 0,96, dan *f1score* 0,96.

4.2.3 Pengujian model CNN dengan pelatihan 500 *epoch*s

Hasil dari pengujian model CNN dengan parameter pelatihan 500 *epochs* disajikan pada *confusion matrix* seperti pada Gambar 4.5 sebagai berikut.



Gambar 4.7 *Confusion Matrix* Pelatihan dengan 500 *Epochs*

Berdasarkan Gambar 4.7 dapat dilihat bahwa data validasi tidak seluruhnya mampu terklasifikasi dengan benar sesuai labelnya. Data berlabel retak buaya ada 55 data yang terklasifikasi dengan benar, 1 data terklasifikasi retak garis, dan 1 data terklasifikasi tidak retak. Data yang berlabel retak garis ada 65 data terklasifikasi dengan benar, 1 data terklasifikasi sebagai kelas retak buaya, dan 2 data terklasifikasi pada kelas tidak retak. Data yang berlabel kelas tidak retak ada 47 data yang terklasifikasi dengan benar, 1 data terklasifikasi pada kelas retak buaya, dan 3 data terklasifikasi pada kelas retak garis.

	precision	recall	f1-score	support
Alligator Crack	0.96	0.96	0.96	57
Linear Crack	0.94	0.96	0.95	68
non crack	0.94	0.92	0.93	51
accuracy			0.95	176
macro avg	0.95	0.95	0.95	176
weighted avg	0.95	0.95	0.95	176

Gambar 4.8 Evaluasi Model CNN dengan Pelatihan 500 *Epochs*

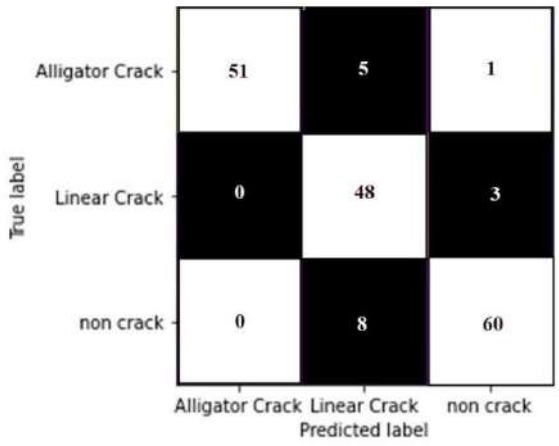
Gambar 4.8 merupakan hasil evaluasi performa *confusion matrix* hasil dari pengujian diperoleh *accuracy* 0.95 *precision* 0,92, *recall* 0,92, dan *f1score* 0,91.

4.3 Hasil Pengujian dengan Variasi *Learning rate*

Pengujian selanjutnya merupakan pengujian model CNN dengan beberapa *learning rate* yang berbeda antara lain 0,01, 0,001, dan 0,0001 pada saat pelatihan. Pengujian ini terdapat 176 gambar yang diujikan.

4.3.1 Pengujian model CNN dengan pelatihan *Learning Rate* 0,01

Hasil dari pengujian model CNN dengan parameter pelatihan *learning rate 0,01* disajikan pada *confusion matrix* seperti pada Gambar 4.9 sebagai berikut.



Gambar 4.9 *Confusion Matrix* Pelatihan dengan *Learning Rate* 0,01

Berdasarkan Gambar 4.9 dapat dilihat bahwa data validasi tidak seluruhnya mampu terklasifikasi dengan benar sesuai labelnya. Data berlabel retak buaya ada 51 data yang terklasifikasi dengan benar, 5 data terklasifikasi retak garis, dan 1 data terklasifikasi tidak retak. Data yang berlabel retak garis ada 48 data terklasifikasi dengan benar, 0 data terklasifikasi sebagai kelas retak buaya, dan 3 data terklasifikasi pada kelas tidak retak. Data yang berlabel kelas tidak retak ada 60 data yang terklasifikasi dengan benar, 0 data terklasifikasi pada kelas retak buaya, dan 8 data terklasifikasi pada kelas retak garis.

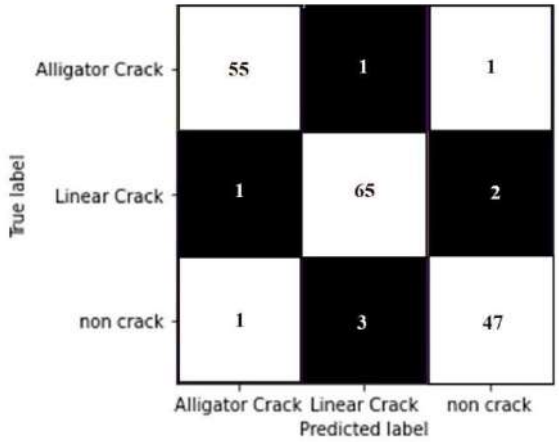
	precision	recall	f1-score	support
retak buaya	1.00	0.89	0.94	57
retak garis	0.79	0.94	0.86	51
tidak retak	0.94	0.88	0.91	68
accuracy			0.90	176
macro avg	0.91	0.91	0.90	176
weighted avg	0.91	0.90	0.91	176

Gambar 4.10 Evaluasi Model CNN dengan Pelatihan *Learning Rate* 0,01

Gambar 4.10 merupakan hasil evaluasi performa *confusion matrix* hasil dari pengujian diperoleh *precision* 0,91, *recall* 0,91, dan *f1-score* 0,90.

4.3.2 Pengujian model CNN dengan pelatihan *learning rate* 0,001

Hasil dari pengujian model CNN dengan parameter pelatihan *learning rate 0,001* disajikan pada *confusion matrix* seperti pada Gambar 4.11 sebagai berikut.



Gambar 4.11 *Confusion Matrix* Pelatihan dengan *Learning Rate* 0,001

Berdasarkan Gambar 4.11 dapat dilihat bahwa data validasi tidak seluruhnya mampu terklasifikasi dengan benar sesuai labelnya. Data berlabel retak buaya ada 55 data yang terklasifikasi dengan benar, 1 data terklasifikasi retak garis, dan 1 data terklasifikasi tidak retak. Data yang berlabel retak garis ada 65 data terklasifikasi dengan benar, 1 data terklasifikasi sebagai kelas retak buaya, dan 2 data terklasifikasi pada kelas tidak retak. Data yang berlabel kelas tidak retak ada 47 data yang terklasifikasi dengan benar, 1 data terklasifikasi pada kelas retak buaya, dan 3 data terklasifikasi pada kelas retak garis. Hasil evaluasi performa *confusion matrix* ditunjukan pada Gambar 4.12.

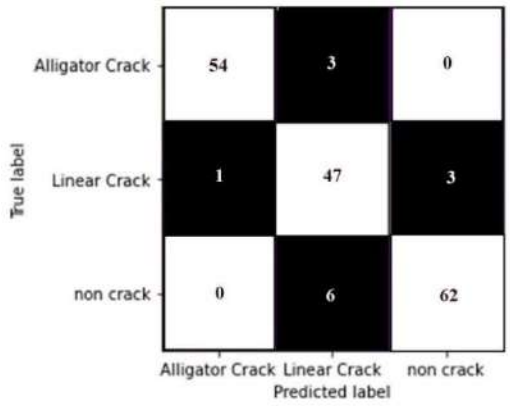
	precision	recall	f1-score	support
Alligator Crack	0.96	0.96	0.96	57
Linear Crack	0.94	0.96	0.95	68
non crack	0.94	0.92	0.93	51
accuracy			0.95	176
macro avg	0.95	0.95	0.95	176
weighted avg	0.95	0.95	0.95	176

Gambar 4.12 Evaluasi Model CNN dengan Pelatihan *Learning Rate* 0,001

Gambar 4.12 merupakan hasil evaluasi performa *confusion matrix* hasil dari pengujian diperoleh *accuracy* 0,95, *precision* 0,95, *recall* 0,95, dan *f1-score* 0,95.

4.3.3 Pengujian model CNN dengan pelatihan *learning rate* 0,0001

Hasil dari pengujian model CNN dengan parameter pelatihan *learning rate 0,0001* disajikan pada *confusion matrix* seperti pada Gambar 4.13 sebagai berikut.



Gambar 4.13 *Confusion Matrix* Pelatihan dengan *Learning Rate* 0,0001

Berdasarkan Gambar 4.13 dapat dilihat bahwa data validasi tidak seluruhnya mampu terklasifikasi dengan benar sesuai labelnya. Data berlabel retak buaya ada 54 data yang terklasifikasi dengan benar, 3 data terklasifikasi retak garis, dan 0 data terklasifikasi tidak retak. Data yang berlabel retak garis ada 47 data terklasifikasi dengan benar, 1 data terklasifikasi sebagai kelas retak buaya, dan 3 data terklasifikasi pada kelas tidak retak. Data yang berlabel kelas tidak retak ada 62 data yang terklasifikasi dengan benar, 0 data terklasifikasi pada kelas retak buaya, dan 6 data terklasifikasi pada kelas retak garis. Hasil evaluasi performa *confusion matrix* ditunjukan pada gambar 4.14.

	precision	recall	f1-score	support
retak buaya	0.98	0.95	0.96	57
retak garis	0.84	0.92	0.88	51
tidak retak	0.95	0.91	0.93	68
accuracy			0.93	176
macro avg	0.92	0.93	0.93	176
weighted avg	0.93	0.93	0.93	176

Gambar 4.14 Evaluasi Model CNN dengan Pelatihan *Learning Rate* 0,0001

Gambar 4.14 merupakan hasil evaluasi performa *confusion matrix* hasil dari pengujian diperoleh *precision* 0,92, *recall* 0,93, dan *f1-score* 0,93.

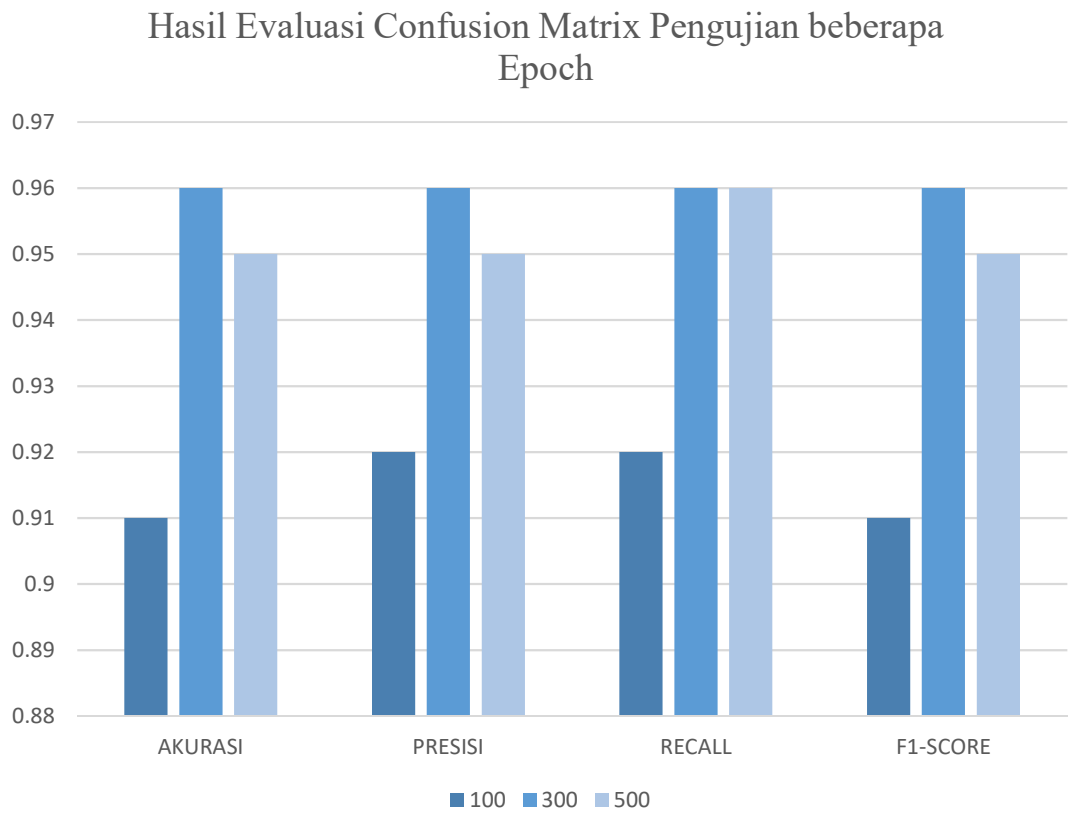
4.4 Perbandingan Hasil Pengujian dengan Variasi *Epoch* dan *Learning rate*

Berdasarkan pengujian yang telah dilakukan maka perbandingan hasil pengujian dengan variasi parameter yang berbeda di sajikan pada Tabel 4.3 untuk perbandingan hasil pengujian dengan variasi *epochs* sebagai berikut.

Tabel 4.3 Perbandingan Jumlah *Epoch* Terhadap Performa Sistem

*Epoch*s	100	300	500
Akurasi	0,91	0,96	0,95
Presisi	0,92	0,96	0,95
Recall	0,92	0,96	0,95
F1-*score*	0,91	0,96	0,95

Tabel 4.3 merupakan perbandingan nilai akurasi, presisi, *recall,* dan *f1score* dengan beberapa jumlah *epoch* yaitu 100, 300, dan 500, Berdasarkan Berdasarkan Tabel 4.3, akurasi terkecil terjadi pada model CNN dengan *training* 100 *epoch*s. Akurasi tertinggi diperoleh oleh model CNN dengan *training* 300 *epochs* sedangkan untuk *precision* terendah diperoleh pada 100 *epoch*s dan *precision* tertinggi diperoleh pada 300 *epochs*. Nilai *recall* dan *f1score* tertinggi dan terendah diperoleh pada model CNN dengan *training epoch* 300 dan 100. Grafik perbandingan hasil pengujian dengan variasi *epochs* disajikan pada Gambar 4.15 sebagai berikut.



Gambar 4.15 Grafik Perbandingan Hasil Pengujian Beberapa *Epoch*

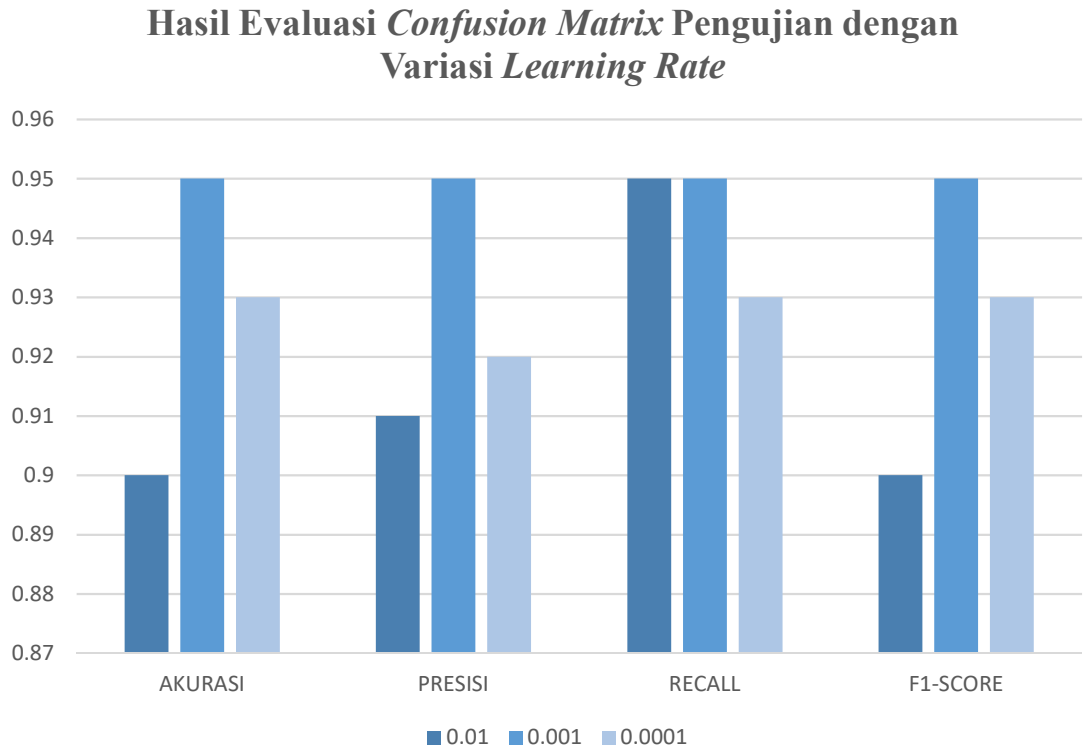
Berdasarkan Gambar 4.15 jumlah *epoch* dapat mempengaruhi performa sistem. Diketahui bahwa semakin banyak jumlah *epoch* yang diberikan membuat akurasi semakin meningkat pada beberapa kasus seperti semakin besar jumlah *epoch* tidak membuat perfoma semakin naik. Hal ini terjadi pada jumlah *epoch* 500, dimana terjadi penurunan nilai akurasi. Terdapat beberapa parameter yang mempengaruhi kondisi tersebut, diantaranya jumlah *filter* sebanyak 32 pada lapisan konvolusi pertama dan kedua serta ukuran *kernel* pada lapisan pertama dan kedua sebesar 3x3.

Berdasarkan pengujian yang telah dilakukan maka perbandingan hasil pengujian dengan variasi parameter yang berbeda disajikan pada Tabel 4.4 untuk perbandingan hasil pengujian dengan variasi *learning rate* sebagai berikut.

Tabel 4.4 Perbandingan Performa Pengujian Variasi *Learning Rate*

Learning rate	**0,01**	**0,001**	**0,0001**
Akurasi	0,90	0,95	0,93
Presisi	0,91	0,95	0,92
Recall	0,91	0,95	0,93
F1-*score*	0,90	0,95	0,93

Berdasarkan Tabel 4.4 penggunaan nilai *learning rate* 0,01 menghasilkan nilai performa yang paling rendah yaitu *accuracy* 0,90, *precision* 0,91, *recall* 0,91 dan *f1score* 0,90 sedangkan untuk performa terbaik diperoleh pada *learning rate* 0,001, yaitu *accuracy* 0,95, *precison* 0,95, *recall* 0,95, dan F1-*score* 0,95. Grafik dari perbandingan hasil pengujian dengan variasi *learning rate* disajikan pada Gambar 4.16 sebagai berikut.



Gambar 4.16 Grafik Perbandingan Hasil Pengujian Beberapa *Learning rate*

Berdasarkan Gambar 4.16, semakin kecil *learning rate*, maka menghasilkan nilai performa yang semakin kecil, seperti pada pengujian ini *learning rate* 0,01

memperoleh *score* performa lebih rendah dibanding *learning rate* 0,001 sedangkan pada *learning rate* 0,0001 mengalami penurunan *score* performa sistem karena semakin kecil nilai *learning rate* memang akan semakin besar *score* performa yang diperoleh, tetapi dibutuhkan waktu pelatihan yang lebih lama untuk memperoleh performa maksimal. Penyebab dari hal ini yaitu terlambatnya proses konvergensi nilai *loss* pada saat proses *training* serta ditinjau dari segi waktu, semakin kecil *learning rate* maka semakin lama waktu yang digunakan untuk *training.*

4.5 Pengujian GUI Sistem (Tampilan Utama Aplikasi)

GUI singkatan dari *Graphical User Interface* merupakan aplikasi visual atau antarmuka yang dapat dikembangkan melalui pemrograman oleh pengguna agar bisa lebih mudah dalam mengatur dan mengontrol alur dari suatu sistem atau aplikasi. Penelitian ini menggunakan *library* `tkinter` dari Python dalam pembuatan GUI untuk deteksi retak jalan dengan metode Wavelet dan CNN. Tampilan utama sistem deteksi retak jalan ditunjukkan pada Gambar lampiran A.2.

4.5.1 Tampilan dan Fungsi Tombol

Tampilan dari GUI beserta tombol-tombol yang ada pada GUI seperti pada Gambar lampiran A.2 tampilan GUI deteksi kerusakan jalan dapat dijelaskan sebagai berikut:

a. Pilih Folder

Tombol ini digunakan untuk memilih folder yang berisikan data citra kondisi jalan retak buaya, retak garis dan tidak retak untuk dijadikan *input*.

b. Pilih transformasi

Tombol ini digunakan untuk mementukan jenis *output* ekstraksi ciri Wavelet yaitu aproximasi, diagonal, horizontal dan vertikal. Ekstraksi ciri tersebut yang kemudian dijadikan *input* untuk proses prediksi dengan model CNN.

c. Proses Gambar

Tombol ini digunakan untuk memulai proses ekstrasi ciri citra menggunakan metode wavelet. Hasil ekstraksi ciri yaitu *output* aproksimasi, horizontal, vertikal, dan

diagonal yang dijadikan *input* dari proses klasifikasi menggunakan metode CNN. Hasil klasifikasi atau prediksi akan ditampilkan dalam tabel seperti pada lampiran 1.8 Gambar GUI deteksi retak jalan raya dan citra asli asli akan ditampilkan dengan keterangan hasil prediksi di atas citra asli.

d. *Next* dan *Previous*

Tombol ini digunakan untuk menganti citra yang ditampilkan hasil transformasi Wavelet ke data cita sebelum atau sesudahnya.

e. Selesai

Tombol ini digunakan untuk berhenti atau keluar dari GUI tampilan dari aplikasi deteksi retak jalan.

4.5.2 Pengujian GUI

Tahap ini adalah proses pengujian terhadap data dan sistem untuk mengetahui kemampuan sistem apakah dapat mendeteksi dan mengklasifikasi objek jalan retak. Pengujian data dilakukan terhadap 17 gambar jalan retak. Hasil dari percobaan dapat dilihat pada tabel 4.5.

Tabel 4.5 Hasil Pengujian Klasifikasi Retak Jalan Raya Menggunakan GUI

No.	Nama	Prediksi	Keterangan
1.	Gambar uji 1	Retak buaya	Benar
2.	Gambar uji 2	Retak buaya	Benar
3.	Gambar uji 3	Retak buaya	Benar
4.	Gambar uji 4	Retak buaya	Benar
5.	Gambar uji 5	Retak buaya	Benar
6.	Gambar uji 6	Tidak retak	Salah
7.	Gambar uji 7	Retak garis	Benar
8.	Gambar uji 8	Retak garis	Benar
9.	Gambar uji 9	Retak garis	Benar
10,	Gambar uji 10	Retak garis	Benar
11.	Gambar uji 11	Retak garis	Benar
12.	Gambar uji 12	Retak garis	Benar
13.	Gambar uji 13	Tidak retak	Benar
14.	Gambar uji 14	Tidak retak	Benar
15.	Gambar uji 15	Tidak retak	Benar
16.	Gambar uji 16	Tidak retak	Benar
17.	Gambar uji 17	Tidak retak	Benar

Berdasarkan tabel 4.5 dapat diperoleh nilai *accuracy* yang dihitung menggunakan Persamaan (2.1) yaitu akurasi sama dengan 94.11%

Nilai *accuracy* dari hasil pengujian di GUI menggunakan *software* Pycharm adalah 94,11% sedangkan nilai *precision* dari pengujian GUI dapat diperoleh dengan perhitungan menggunakan Persamaan (2.2) memperoleh hasil dari dari pengujian GUI memperoleh nilai *precision* 0.83 untuk retak buaya, 1 untuk retak garis dan 1 untuk tidak retak, sehingga rata-rata *precision* dari pengujian GUI adalah 0,943.

Nilai *recall* dari pengujian GUI dapat diperoleh dengan perhitungan menggunakan Persamaan (2.4) memperoleh nilai *recall* 1 untuk retak buaya, 1 untuk retak garis dan 0,83 untuk tidak retak, sehingga rata-rata *recall* dari pengujian GUI adalah 0,943.

Nilai *f1score* dari pengujian GUI dapat diperoleh dengan perhitungan menggunakan Persamaan (2.6) diperoleh nilai yang didapat dari perhitungan *f1-score* di atas adalah 0,943. Nilai *f1-score* dapat dipertimbangkan menjadi acuan performa model klasifikasi apabila *false negative* dan *false positive* data klasifikasi nilanya tidak saling mendekati. Nilai *f1-score* adalah 0,943 dan nilai *accuracy* adalah 94,11%, keduanya saling mendekati dan dapat menjadi acuan performa klasifikasi retak jalan raya.

Nilai yang didapatkan dari klasifikasi ini sudah mendekati angka 100%. Penelitian ini dilakukan semaksimal mungkin untuk mencapai nilai yang paling optimal dan dapat disimpulkan bahwa pengujian GUI untuk deteksi retak jalan berjalan dengan baik.

4.6 Pengujian Deteksi Retak Jalan Raya Secara *Real-time*

Tahap ini adalah proses pengujian terhadap data dan sistem untuk mengetahui kemampuan sistem apakah dapat mendeteksi dan mengklasifikasi objek jalan retak secara *real-time*. Pengujian data dilakukan terhadap sebuah video dan diperoleh 14 gambar retak jalan yang tangkap dari video tersebut. Hasil dari percobaan dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Pengujian Klasifikasi Retak Jalan Raya Secara *Real-time*

No.	Nama	Prediksi	Keterangan
1.	Video Capture 1	Retak garis	Benar
2.	Video Capture 2	Retak buaya	Benar
3.	Video Capture 3	Retak buaya	Benar
4.	Video Capture 4	Retak buaya	Benar
5.	Video Capture 5	Retak buaya	Benar
6.	Video Capture 6	Retak buaya	Benar
7.	Video Capture 7	Retak buaya	Benar
8.	Video Capture 8	Retak buaya	Benar
9.	Video Capture 9	Retak buaya	Benar
10,	Video Capture 10	Retak garis	Benar
11.	Video Capture 11	Retak garis	Benar
12.	Video Capture 12	Retak garis	Benar
13.	Video Capture 13	Retak garis	Benar
14.	Video Capture 14	Retak garis	Benar
15.	Video Capture 15	Tidak retak	Salah
16.	Video Capture 16	Retak garis	Benar

Berdasarkan Tabel 4.6 diperoleh hasil nilai akurasi dari hasil pengujian di deteksi *real-time* menggunakan YOLO yang dihitung menggunakan Persamaan (2.1) diperoleh hasil nilai akurasi dari hasil pengujian di deteksi *real-time* menggunakan YOLO adalah 93,75%. Nilai precision dari pengujian GUI dapat diperoleh dengan perhitungan menggunakan Persamaan (2.2) memperoleh hasil dari dari pengujian klasifikasi retak secara *real-time* memperoleh nilai *precision* 1 untuk retak buaya dan 0,875 untuk retak garis, sehingga rata-rata *recall* dari pengujian GUI adalah 0,937 sehingga nilai recall dari pengujian GUI dapat diperoleh dengan perhitungan menggunakan Persamaan (2.4). Hasil dari dari pengujian klasifikasi retak secara *real-time* memperoleh nilai *recall* 1 untuk retak buaya dan 0,875 untuk retak garis, sehingga rata-rata *recall* dari pengujian GUI adalah 0,9375.

Nilai *f1score* dihitung menggunakan Persamaan (2.6). Berdasarkan perhitungan menggunakan Persamaan (2.6) diperoleh nilai *recall* yang didapat dari perhitungan *f1Score* di atas adalah 94,3%. *Nilai f1score* dapat dipertimbangkan menjadi acuan performa model klasifikasi. Hal ini bisa dilakukan apabila *false negative* dan *false positive* data klasifikasi nilanya tidak saling mendekati. Nilai *F1 Score* adalah 100% dan nilai akurasi adalah 100%, keduanya sama dan dapat menjadi acuan performa klasifikasi retak jalan raya.

Nilai yang didapatkan dari klasifikasi ini sudah mendekati angka 100% sehingga deteksi retak jalan raya secara *real-time* untuk deteksi retak jalan berjalan dengan baik.

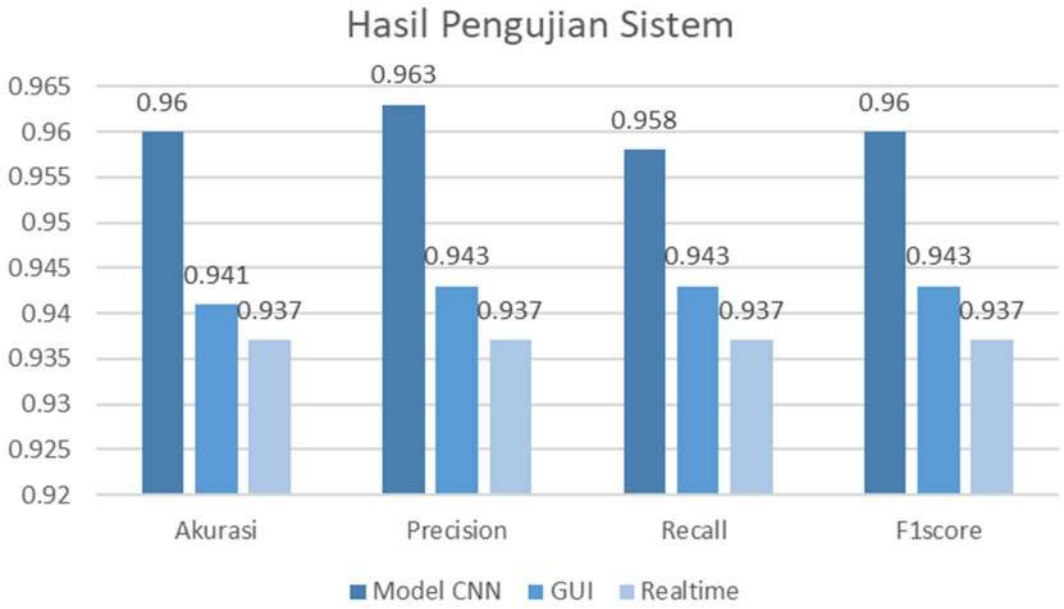
4.7 Hasil Pengujian Performa Sistem

Performa sistem deteksi retak jalan raya dapat dilihat dengan menghitung nilai *accuracy, precision, recall* dan *F1score*. Tabel 4.7 menunjukan hasil pengujian performa sistem deteksi retak.

Tabel 4.7 Hasil Pengujian Performa Sistem

No	Pengujian	Akurasi	*Precision*	*Recall*	F1score
1	Model CNN	0,960	0,963	0,958	0,960
2	GUI	0,941	0,943	0,943	0,943
3	*Real-time*	0,937	0,937	0,937	0,937

Berdasarkan Tabel 4.7 dari hasil pengujian di atas, pengujian deteksi retak secara *real-time* mendapatkan hasil paling unggul dengan akurasi 93,75%. Evaluasi kinerja model dalam mengklasifikasi retak jalan raya secara *real-time* mendapatkan nilai *precision* sebesar 93,75%, nilai *recall* sebesar 93,75% dan *f1score* sebesar 93,75% sedangkan untuk pengujian pada GUI memperoleh hasil dengan nilai akurasi 94,11%, nilai *precision* sebesar 0,943, nilai *recall* sebesar 0,9433 dan *f1score* sebesar 0,943. Pengujian data validasi di Google *Colaboratory* memperoleh hasil akurasi 96,59%, nilai *precision* sebesar 0,966, nilai *recall* sebesar 0,963 dan *f1score* sebesar 0,964. Berikut grafik hasil pengujian sistem ditunjukkan oleh Gambar 4.18.



Gambar 4.18 Grafik Hasil Pengujian

Berdasarkan Gambar 4.18 hasil pengujian sistem dapat mengklasifikasikan jalan retak menggunakan *notebook* Google *Colaboratory*, menggunakan GUI, dan secara *real-time* menggunakan *input* sebuah video dengan baik. Kesalahan sistem dalam mengklasifikasikan beberapa retak karena jumlah *dataset* yang kurang banyak dan kualitas data *dataset* yang kurang baik karena kondisi lapangan dan citra yang berbayang karena di ambil pada siang hari sehingga pengambilan ekstraksi ciri tidak optimal.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, pada Implementasi metode Wavelet dan *backpropagation neural network* pada deteksi retak jalan raya berbasis pengolahan citra, maka dapat diambil kesimpulan sebagai berikut:

1. Metode CNN dan YOLO mampu mendeteksi retak jalan raya dengan akurat. Proses ektraksi ciri dan *deep learning* dari data masukan berupa citra kondisi jalan mambuat sistem mampu mendekteksi citra retak buaya, retak garis dan tidak retak.
2. Hasil *training* data pada metode *deep learning* dipengaruhi oleh parameter *epoch* dan *learning rate*. Semakin banyak jumlah *epoch* yang digunakan, maka nilai akurasi yang dihasilkan akan semakin tinggi, namun dapat menyebabkan terjadinya *overfitting*. Sementara itu, semakin kecil nilai *learning rate* yang digunakan, maka nilai akurasi yang diperoleh akan semakin tinggi, namun membutuhkan waktu yang lambat untuk mencapai konvergensi.
3. Pengujian model CNN di Google *Colaboratory* memperoleh hasil akurasi 0,960, nilai *precision* sebesar 0,963, nilai *recall* sebesar 0,958 dan F1score sebesar 0,960. Pengujian *real-time* mendapatkan nilai *precision* sebesar 0,937, nilai *recall* sebesar 0,937 dan *F1score* sebesar 0,937. Pengujian GUI memperoleh hasil terendah dengan nilai akurasi 0,941 nilai *precision* sebesar 0,943, nilai *recall* sebesar 0,943 dan F1score sebesar 0,943.

5.2 Saran

Setelah melakukan penelitian pada Implementasi metode Wavelet dan *backpropagation neural network* pada deteksi retak jalan raya berbasis pengolahan citra, saran yang ingin disampaikan sebagai berikut:

1. Penelitian selanjutnya dapat menambahkan jumlah *dataset* yang digunakan dalam proses pelatihan sehingga mencapai hasil akurasi yang tinggi.

2. Penelitian selanjutnya dapat menggunakan metode klasifikasi citra lainnya seperti *K-Neighborhood Neighbors* (KNN), *Support Vector Machine* (SMV), dan *Linier Discriminant Analysis* (LDA).
3. Penelitian selanjutnya dapat menambah jenis kerusakan jalan lainnya seperti lubang.
4. Penelitian selanjutnya dapat membuat GUI untuk deteksi secara *real-time* dan menambahkan GPS sebagai instrumentasi penelitian untuk mendapatkan detail lokasi keretakan jalan.

DAFTAR PUSTAKA

[1] H. Oliveira and P. L. Correia, "Road surface *crack* detection: Improved segmentation with pixel-based refinement," in 2017 25th European Signal Processing Conference (EUSIPCO), pp. 2026–2030, 2017.

[2] N. A. Munggarani And A. Wibowo, "Kajian Faktor-Faktor Penyebab Kerusakan Dini Perkerasan Jalan Lentur Dan Pengaruhnya Terhadap Biaya Penanganan," Jurnal Infrastruktur, Vol. 3, No. 01, P. 10, 2017.

[3] S. Bang, S. Park, H. Kim, Y. Yoon, and H. Kim, "A Deep Residual Network with Transfer Learning for Pixel-level Road Crack Detection," presented at the 34th International Symposium on Automation and Robotics in Construction, Taipei, Taiwan, Jul. 2018.

[4] R. Medina, J. Llamas, E. Zalama, and J. Gomez-Garcia-Bermejo, "Enhanced automatic detection of road surface *crack*s by combining 2D/3D image processing techniques," in 2014 IEEE International Conference on Image Processing (ICIP), pp. 778–782, 2014.

[5] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road *crack* detection using deep Convolution neural network," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3708–3712. 2016.

[6] V. Mandal, L. Uong, and Y. Adu-Gyamfi, "Automated Road Crack Detection Using Deep Convolution Neural Networks," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5212–5215, 2018.

[7] B. Pitaloka, B. Pitaloka, A. Putra, and S. Lestari, "Implementasi Metode Forward Chaining Dan Backward Chaining Dalam Mendeteksi Kerusakan Pada Prasarana Lalu Lintas," 2023.

[8] D. Ma, H. Fang, N. Wang, B. Xue, J. Dong, and F. Wang, "A real-time *crack* detection algorithm for pavement based on CNN with multiple feature *layer*s," *Road Materials and Pavement Design*, vol. 23, no. 9, pp. 2115–2131, Sep. 2022.

[9] S. A. Shifani, P. Thulasiram, K. Narendran, and D. R. Sanjay, "A Study of Methods using Image Processing Technique in Crack Detection," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pp. 578–582, 2020,

[10] A. N. Utomo, "*Extraction Data Analysis Of Histogram Moment Image And Comparison Of Classification Algorithm Models Of Naive Bayes, Nearest Neighbor, Support Vector Machine, And Decision Tree In Case Study Of Damaged Asphalt And Undamaged Asphalt Road Images*," vol. 9, 2020.

[11] N. Neneng, A. S. Puspaningrum, and A. A. Aldino, "Perbandingan Hasil Klasifikasi Jenis Daging Menggunakan Ekstraksi Ciri Tekstur Gray Level Co-occurrence Matrices (GLCM) Dan Local Binary Pattern (LBP)," *SMATIKA*, vol. 11, no. 01, pp. 48–52, Jul. 2021.

[12] F. M. Sarimole and M. I. Fadillah, "Classification Of Guarantee Fruit Murability Based on HSV Image With K-Nearest Neighbor," *JAETS*, vol. 4, no. 1, pp. 48–57, Sep. 2022.

[13] D. S. Guru, "Texture Features and KNN in Classification of Flower Images".

[14] N. Rasiwasia and N. Vasconcelos, "Latent Dirichlet Allocation Models for Image Classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2665–2679, Nov. 2013.

[15] P. Zhu, J. Isaacs, B. Fu, and S. Ferrari, "Deep learning feature extraction for target recognition and classification in underwater sonar images," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Melbourne, Australia: IEEE, pp. 2724–2731, 2017.

[16] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple Convolution neural network on image classification," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA),* pp. 721–724, 2017.

[17] N. Shatnawi, "Automatic Pavement Cracks Detection using Image Processing Techniques and Neural Network," ijacsa, vol. 9, no. 9, 2018.

[18] S. Riyadi, F. Yusfida A'la, C. Oktomy, and K. Hawari Ghazali, "Road Surface Crack Detection using Wavelets Features Extraction Technique," Indian Journal of Science and Technology, vol. 9, no. 47, Dec. 2016.

[19] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, "Convolution Neural Network (CNN) for Image Detection and Recognition," in 2018 First International Conference on Secure Cyber Computing and Communication 61 Universitas Sultan Ageng Tirtayasa (ICSCCC), Jalandhar, India, Dec. 2018, pp. 278–282..

[20] S. Singha and B. Aydin, "Automated Drone Detection Using YOLOv4," Drones, vol. 5, no. 3, p. 95, Sep. 2021.

[21] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767 [cs], Apr. 2018, Accessed: Mar. 30, 2022. [Online]. Available: http://arxiv.org/abs/1804.02767

[22] Pemerintah Indonesia, "Undang - Undang Republik Indonesia No. 22 Tahun 2009 Tentang Lalu Lintas dan Angkutan Jalan. Lembaran Negara Republik Indonesia Tahun 2009, No. 96," Sekretariat Negara, Jakarta, 2009.

[23] Shahin, M. Y. (1994), Pavement management for airports, roads, and parking lots.

[24] Ikhwanul, F. Yudaningrum, "Analisis Kerusakan Retak Padar Ruas Jalan Kedungmundu-Metesih Serta Metode Perbaikannya," Prosiding Seminar Nasional, ISBN: 978-602-14020-3-0, 2016.

[25] Putra, Darma, (2010). Pengolahan Citra Digital. Yogyakart: Penerbit Andi.

[26] W. S. Eka Putra, "Klasifikasi Citra Menggunakan Convolution Neural Network (CNN) pada Caltech 101," *JTITS*, vol. 5, no. 1, Mar. 2016.

[27] Redmon, J., Divvala, S., Girshick, R. dan Farhadi, A., You only look once: Unifed, real-time object detection, in `Proceedings of the IEEE Conference on computer vision and pattern recognition', pp. 779-788, 2016.

[28] 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, 2023.

[29] S. Jana, S. Thangam, A. Kishore, V. Sai Kumar, and S. Vandana, "Transfer learning based deep Convolution neural network model for pavement *crack* detection from images," *IJNAA*, vol. 13, no. 1, Jan. 2022.

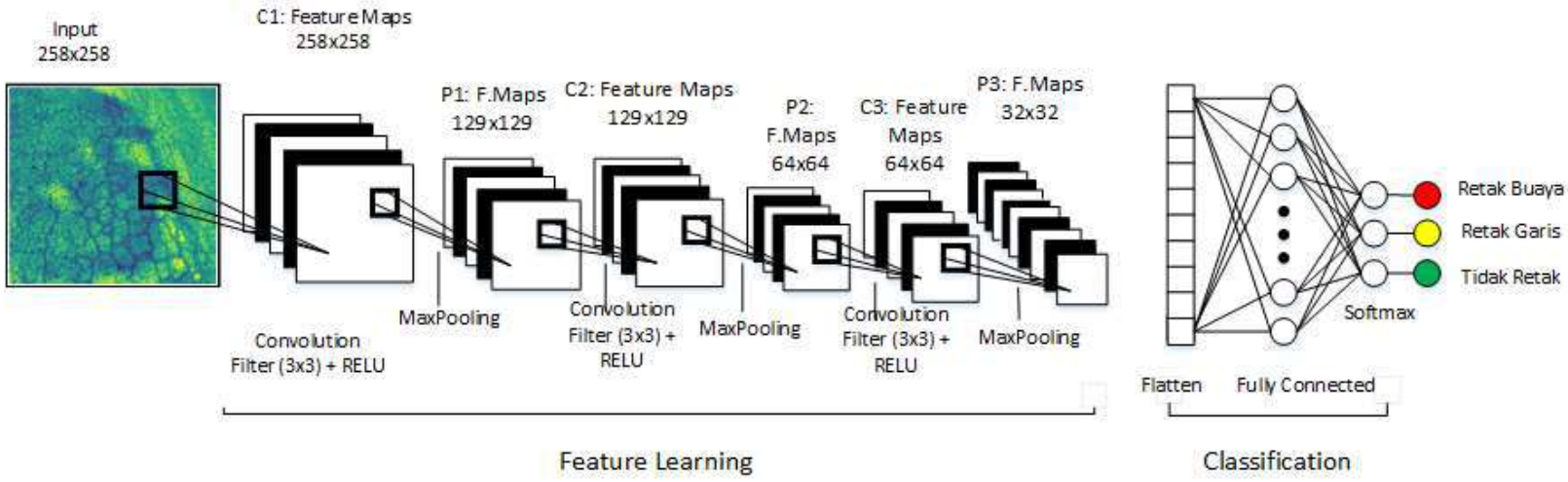
[30] N. Safaei, O. Smadi, A. Masoud, and B. Safaei, "An Automatic Image Processing Algorithm Based on Crack Pixel Density for Pavement Crack Detection and Classification," *Int. J. Pavement Res. Technol.*, vol. 15, no. 1, pp. 159–172, Jan. 2022.

[31] S. Liu, Y. Han, and L. Xu, "Recognition of road *crack*s based on multi-scale Retinex fused with wavelet transform," *Array*, vol. 15, p. 100193, Sep. 2022.

[32] A. Ashraf, A. Sophian, A. A. Shafie, T. S. Gunawan, N. N. Ismail, and A. A. Bawono, "Detection of Road Cracks Using Convolution Neural Networks and Threshold Segmentation," *JIAE*, vol. 2, no. 2, pp. 123–134, Sep. 2022.

LAMPIRAN

LAMPIRAN A Arsitektur CNN



Lampiran A.1 Gambar Arsitektur CNN

Lampiran A. 2 Tampilan GUI

Lampiran B *Listing Code*

```
import os
import pickle
from tqdm import tqdm

import tkinter as tk
from tkinter.constants import ANCHOR, BOTTOM, CENTER, INSERT, LEFT,
RIGHT, S, TOP, X
from tkinter import ttk, filedialog

import numpy as np
import pandas as pd
from pandastable import Table

import cv2
import pywt
from PIL import Image, ImageTk

from tensorflow import keras

# ==== Constants ==== #
BACKGROUND = "#4bacc6"
ALTBACKGROUND = "#4bacb0"
shape = (512,512)
le = pickle.load(open(os.path.join("model","le.sav"),"rb"))
model = keras.models.load_model('model')
# ==== Constants ==== #

# ==== GUI ==== #
class root(tk.Tk):
    def __init__(self):
        # head
        super(root, self).__init__()
        self.title("Deteksi Keretakan Jalan")
        self.geometry("1024x768")
        self['background'] = BACKGROUND
        # top frame
        self.header = tk.Frame(self, background=BACKGROUND)
        self.header.pack(side=TOP)
        self.headerL = tk.Frame(self.header, background=BACKGROUND)
        self.headerL.pack(side=LEFT)
        self.headerR = tk.Frame(self.header, background=BACKGROUND)
        self.headerR.pack(side=LEFT)
        self.top = tk.Frame(self.headerR, background=BACKGROUND)
        self.top.pack(side=TOP)

        self.logo =
Image.open(os.path.join("assets","logo.png")).resize((150,150))
        self.logotk = ImageTk.PhotoImage(image=self.logo)
        self.logo_frame = tk.Label(self.headerL, image= self.logotk,
```

```
background=BACKGROUND)
        self.logo_frame.pack(side=LEFT)

        self.label = tk.Label(self.top, text='Deteksi Keretakan
Jalan', fg='black', bg='#4bacc6', padx=5, pady=5)
        self.label.pack(anchor='n', side='top', padx=20, pady=3)
        self.label.config(font=("Comic Sans", 32, 'bold'))

        # define frame
        self.sub = tk.Frame(self.headerR, background=BACKGROUND,
padx=10, pady=3)
        self.sub.pack(side=TOP, pady=3)
        self.subL = tk.Frame(self.sub, background=BACKGROUND,
padx=10, pady=3)
        self.subL.pack(side=LEFT, pady=3)
        self.subC = tk.Frame(self.sub, background=BACKGROUND,
padx=10, pady=3)
        self.subC.pack(side=LEFT, pady=3)
        self.subR = tk.Frame(self.sub, background=BACKGROUND,
padx=10, pady=3)
        self.subR.pack(side=LEFT, pady=3)
        self.bottom = tk.Frame(self, background=BACKGROUND, padx=10,
pady=3)
        self.bottom.pack(side=TOP)
        self.frame = tk.Frame(self, background=BACKGROUND, padx=10,
pady=3)
        self.frame.pack(side=TOP, pady=3)
        self.frameL = tk.Frame(self.frame, background=BACKGROUND,
padx=10, pady=3)
        self.frameL.pack(side=LEFT, pady=3)
        self.frameR = tk.Frame(self.frame, background=BACKGROUND,
padx=10, pady=3)
        self.frameR.pack(side=LEFT, pady=3)
        self.frTop = tk.Frame(self.frameR, background=BACKGROUND,
padx=10, pady=3)
        self.frTop.pack(side=TOP, pady=3)
        self.frMid = tk.Frame(self.frameR, background=BACKGROUND,
padx=10, pady=3)
        self.frMid.pack(side=TOP, pady=3)
        self.frBtm = tk.Frame(self.frameR, background=BACKGROUND,
padx=10, pady=3)
        self.frBtm.pack(side=BOTTOM, pady=3)

        self.folderPath = tk.StringVar()
        self.inputlabel = tk.Label(self.subL,text="Masukkan Folder")
        self.inputlabel.pack(side=LEFT, pady=10)
        self.inputtext =
tk.Entry(self.subC,textvariable=self.folderPath, width=50)
        self.inputtext.pack(side=LEFT, pady=10)
        self.btnFind = ttk.Button(self.subR, text="Browse
```

```
Folder",command=self.folderpath)

          self.btnFind.pack(side=LEFT, pady=10)
        self.options_list = ["Approximation", "Horizontal Detail",
"Vertical Detail", "Diagonal Detail"]
        self.value_inside = tk.StringVar()
        self.value_inside.set("Pilih Transformasi")

        self.button1 = tk.Button(self.bottom, text="Selesai",
command=self._quit)
        self.button1.pack(side=RIGHT, pady=2, padx=10)
        self.button2 = tk.Button(self.bottom, text="Proses Gambar",
command=self.doStuff)
        self.button2.pack(side=LEFT, pady=2, padx=10)

        self.question_menu = tk.OptionMenu(self.bottom,
self.value_inside, *self.options_list)
        self.question_menu.pack(side=TOP, pady=2, padx=10)

        self.imgfilename = tk.StringVar()

        self.num = tk.IntVar()
    def _quit(self):
        self.quit()
        self.destroy()

    def folderpath(self):
        self.folder_selected = filedialog.askdirectory()
        self.folderPath.set(self.folder_selected)

    def doStuff(self):
        if isinstance(self.num,int):
            self.filename.destroy()
            self.resultlabel.destroy()
            self.image_frame.destroy()
            self.prev.destroy()
            self.next.destroy()
        root_folder = self.folderPath.get()
        transform = self.value_inside.get()
        print("Doing stuff with folder", root_folder, transform)
        self.image_path = []
# membuat list kosong untuk lokasi image
        for img in os.listdir(root_folder):
# loop setiap folder yang ada dalam folder data
            self.image_path.append(os.path.join(root_folder,img))
# tambahkan path gambar ke list image_path
        self.image = []
# membuat list kosong untuk image array
        self.imageH = []
# membuat list kosong untuk image array
        self.imageV = []
```

```
# membuat list kosong untuk image array
        self.imageD = []

# membuat list kosong untuk image array
        for img in tqdm(self.image_path):
# loop setiap image path
            arr = cv2.imread(img)
# membaca image
            arr = cv2.resize(arr,shape)
# resize image
            arr = cv2.cvtColor(arr, cv2.COLOR_BGR2LAB)
# convert BGR ke CIE Lab
            l, a, b = cv2.split(arr)
# split image channels
            clahe = cv2.createCLAHE(clipLimit=4.0,
tileGridSize=(8,8))                                                    #
membuat fungsi adaptive histogram equalization, untuk memperbaiki
contrast
            cl = clahe.apply(l)
# apply fungsi adaptve histogram equalization terhadap channel l
            arr = cv2.merge((cl,a,b))
# merge kembali ketiga channle gambar
            arr = cv2.cvtColor(arr, cv2.COLOR_LAB2BGR)
# convert CIE Lab ke BGR
            arr = cv2.cvtColor(arr,cv2.COLOR_BGR2GRAY)
# convert ke grayscale
            coeffs2 = pywt.dwt2(arr, 'bior1.3')
# transformasi Wavelet
            LL, (LH, HL, HH) = coeffs2
# membuat variable untuk setiap approximation
            self.image.append(LL.tolist())
# add hasil transformasi Wavelet ke list image
            self.imageH.append(LH.tolist())
# add hasil transformasi Wavelet ke list image
            self.imageV.append(HL.tolist())
# add hasil transformasi Wavelet ke list image
            self.imageD.append(HH.tolist())
# add hasil transformasi Wavelet ke list image
        if transform == "Horizontal Detail":
            X = np.array(self.imageH)/255
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
        elif transform == "Vertical Detail":
            X = np.array(self.imageV)/255
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
        elif transform == "Diagonal Detail":
            X = np.array(self.imageD)/255
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
```

```
else:
    X = np.array(self.image)/255
```

```
# membuat variable X berisi image array yang telah di-transformed
dan di normalisasi
        input_shape = (X.shape[1],X.shape[2])

# membuat variable input_shape sebagai input shape model
        X = X.reshape(-1, input_shape[0], input_shape[1], 1)
# reshape X
        predictions = model.predict(X)
# membuat prediksi terhadap data
        predictions = np.argmax(predictions, axis=-1)
# konversi hasil prediksi sesuai dengan bentuk array mula-mula
        self.results = le.inverse_transform(predictions)
# convert ke label semula
        df = pd.DataFrame({
            "image":self.image_path,
            "prediction":self.results})
        df['image'] =
df['image'].str.split("/",expand=True).iloc[:,-1]
        self.table = pt = Table(self.frameL, dataframe=df,
                                    showtoolbar=True,
showstatusbar=True, width=500)
        pt.show()
        self.num = 0
        self.imgfilename = self.image_path[self.num].split("/")[-1]
        self.filename = tk.Label(self.frTop, text=self.imgfilename,
fg='black', bg='white', padx=5, pady=5)
        self.filename.pack(side=TOP, pady=2, padx=10)
        self.resultlabel = tk.Label(self.frTop,
text=self.results[self.num], fg='black', bg='white', padx=5, pady=5)
        self.resultlabel.pack(side=TOP, pady=2, padx=10)
        self.img =
Image.open(self.image_path[self.num]).resize((300,300))
        self.imgtk = ImageTk.PhotoImage(image=self.img)
        self.image_frame = tk.Label(self.frMid, image= self.imgtk)
        self.image_frame.pack(side=LEFT)
        self.prev = tk.Button(self.frBtm, text="Previous",
command=self._previous)
        self.prev.pack(side=LEFT, pady=2, padx=10)
        self.next = tk.Button(self.frBtm, text="Next",
command=self._next)
        self.next.pack(side=LEFT, pady=2, padx=10)
    def _next(self):
        if self.num == len(self.image_path):
```

```
    self.num = self.num
        else:
            self.num+=1
        self.filename.destroy()
        self.resultlabel.destroy()
        self.image_frame.destroy()
        self.imgfilename = self.image_path[self.num].split("/")[-1]
        self.filename = tk.Label(self.frTop, text=self.imgfilename,
fg='black', bg='white', padx=5, pady=5)
        self.filename.pack(side=TOP, pady=2, padx=10)
        self.resultlabel = tk.Label(self.frTop,
text=self.results[self.num], fg='black', bg='white', padx=5, pady=5)
        self.resultlabel.pack(side=TOP, pady=2, padx=10)

self.img = Image.open(self.image_path[self.num]).resize((300,300))
        self.imgtk = ImageTk.PhotoImage(image=self.img)
        self.image_frame = tk.Label(self.frMid, image= self.imgtk)
        self.image_frame.pack(side=LEFT)
    def _previous(self):
        if self.num == 0:
            self.num = self.num
        else:
            self.num-=1
        self.filename.destroy()
        self.resultlabel.destroy()
        self.image_frame.destroy()
        self.imgfilename = self.image_path[self.num].split("/")[-1]
        self.filename = tk.Label(self.frTop, text=self.imgfilename,
fg='black', bg='white', padx=5, pady=5)
        self.filename.pack(side=TOP, pady=2, padx=10)
        self.resultlabel = tk.Label(self.frTop,
text=self.results[self.num], fg='black', bg='white', padx=5, pady=5)
        self.resultlabel.pack(side=TOP, pady=2, padx=10)
        self.img =
Image.open(self.image_path[self.num]).resize((300,300))
        self.imgtk = ImageTk.PhotoImage(image=self.img)
        self.image_frame = tk.Label(self.frMid, image= self.imgtk)
        self.image_frame.pack(side=LEFT)

road = root()
road.mainloop()
# ==== GUI ==== #
```

Lampiran C Citra Pengujian

Tabel C.1 Gambar retak jalan raya untuk pengujian GUI

No	Foto kondisi jalan	Keterangan
1		Retak Buaya
2		Retak Buaya
3		Retak Buaya

4		Retak Buaya
5		Retak Buaya
6		Retak Buaya
7		Retak Garis

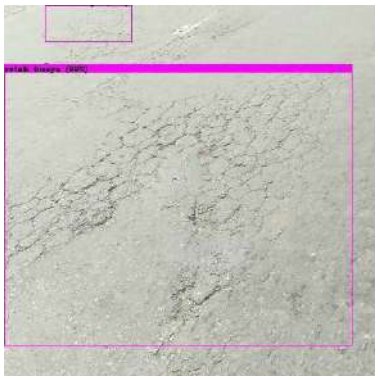
8		Retak Garis
9		Retak Garis
10		Retak Garis
11		Retak Garis

12		Retak Garis
13		Tidak Retak
14		Tidak Retak
15		Tidak Retak

16		Tidak Retak
17		Tidak Retak

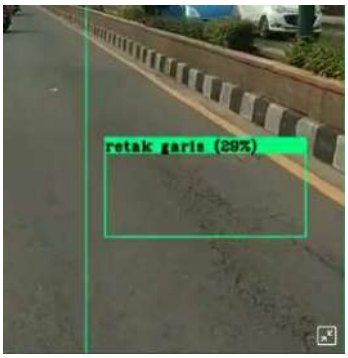
Tabel B.2 Gambar retak jalan raya untuk pengujian YOLO

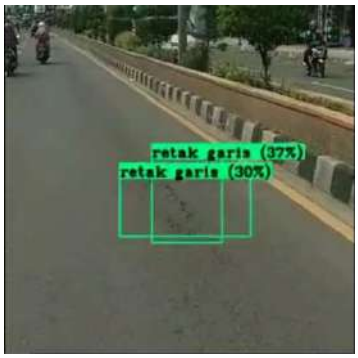
No	Foto kondisi jalan	Keterangan
1		Retak Garis

2		Retak Buaya
3		Retak Buaya
4		Retak Buaya

5		Retak Buaya
6		Retak Buaya
7		Retak Buaya

8		Retak Buaya
9		Retak Buaya
10		Retak Garis

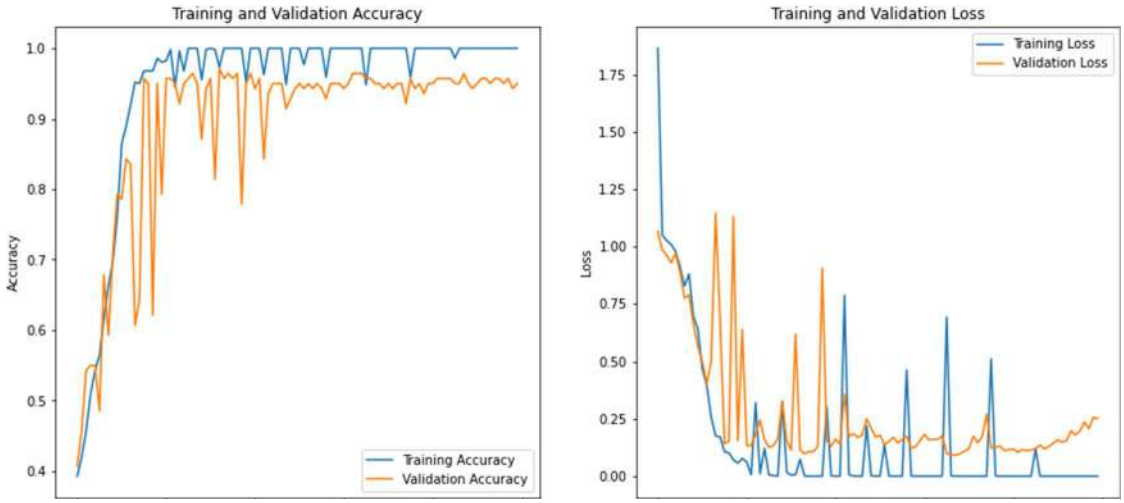
11		Retak Garis
12		Retak Garis
13		Retak Garis
14		Retak Garis

15		Retak Buaya
16		Retak Garis

LAMPIRAN D Pelatihan CNN

D.1 Pelatihan dengan variasi *epoch*

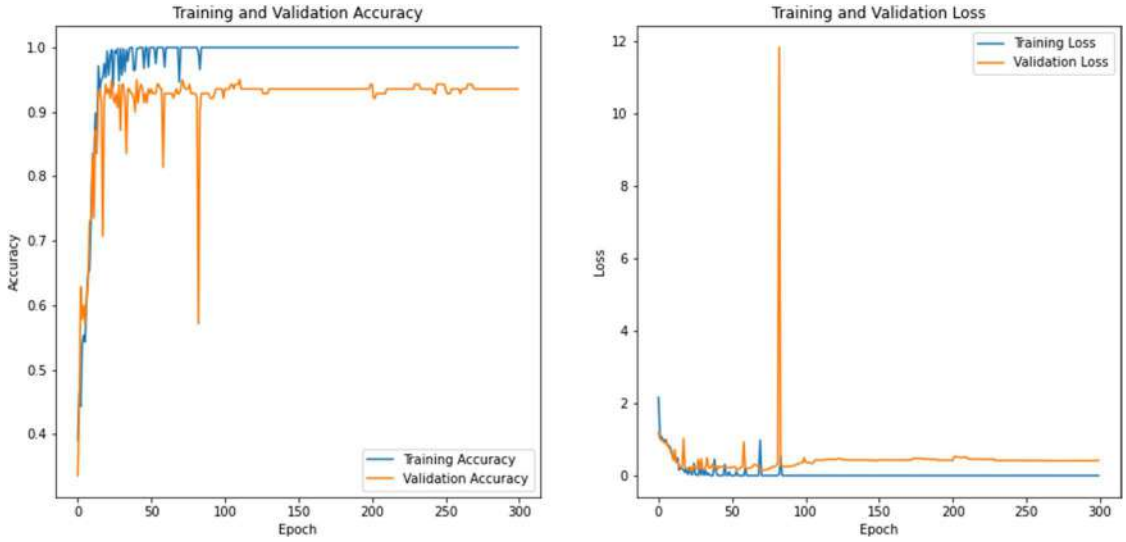
Dari grafik yang terdapat pada Gambar D.1, kita dapat melihat perkembangan *loss* dan akurasi selama pelatihan dengan total 100 *epoch*. Hasilnya menunjukkan bahwa akurasi pelatihan mencapai 100%, sedangkan akurasi validasi mencapai 95% setelah melewati 100 *epoch* dengan *learning rate* 0,001. Di sini, "*epoch*" merupakan parameter yang mengindikasikan jumlah iterasi penuh terhadap seluruh dataset yang mengalami proses pelatihan dalam model arsitektur *deep learning*. Dengan pengaturan 100 *epoch*, seluruh dataset telah menjalani pelatihan sebanyak 100 kali. Sementara itu, *loss* pelatihan mencapai 0,00, dan *loss* validasi mencapai 0,251..



Gambar D.1 Grafik Hasil *Training Validation Epoch* 100

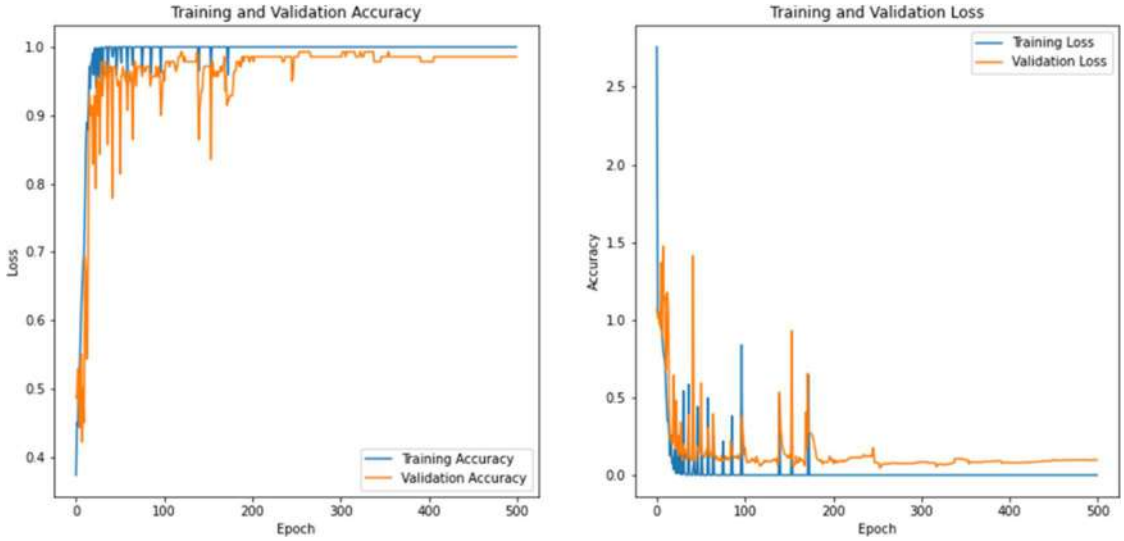
Gambar D.2 menunjukan grafik *loss* dan akurasi pelatihandengan *epoch* 300 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 93,57 % setelah melalui 300 *epochs* dengan *learning rate* 0,001. *Epoch* adalah parameter untuk seluruh dataset yang sudah melalui proses pelatihan pada model arsitektur *deep learning* untuk sekali putaran penuh, dan jika mengatur dengan 100 *epoch* artinya seluruh dataset mengalami proses pelatihan sebanyak 100 kali.

Sementara untuk *training loss* mencapai 0,00 dan *validation loss* mencapai 0,421.



Gambar D.2 Grafik Hasil *Training Validation Epoch* 300

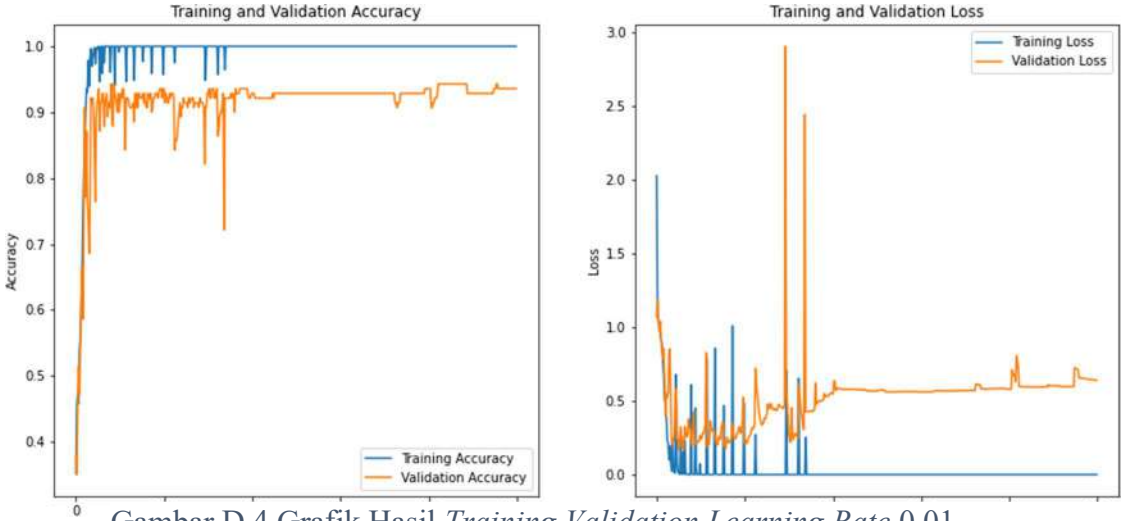
Gambar D.3 gambar grafik *loss* dan akurasi pelatihandengan *epoch* 500 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 96,43 % setelah melalui 500 *epochs* dengan *learning rate* 0,001. *Epoch* adalah parameter untuk seluruh dataset yang sudah melalui proses pelatihan pada model arsitektur *deep learning* untuk sekali putaran penuh, dan jika mengatur dengan 100 *epoch* artinya seluruh dataset mengalami proses pelatihan sebanyak 100 kali. Sementara untuk *training loss* mencapai 0,00 dan *validation loss* mencapai 0,3636.



Gambar D.3 Grafik Hasil *Training Validation Epoch* 500

D.2 Pelatihan Dengan Variasi *Learning rate*

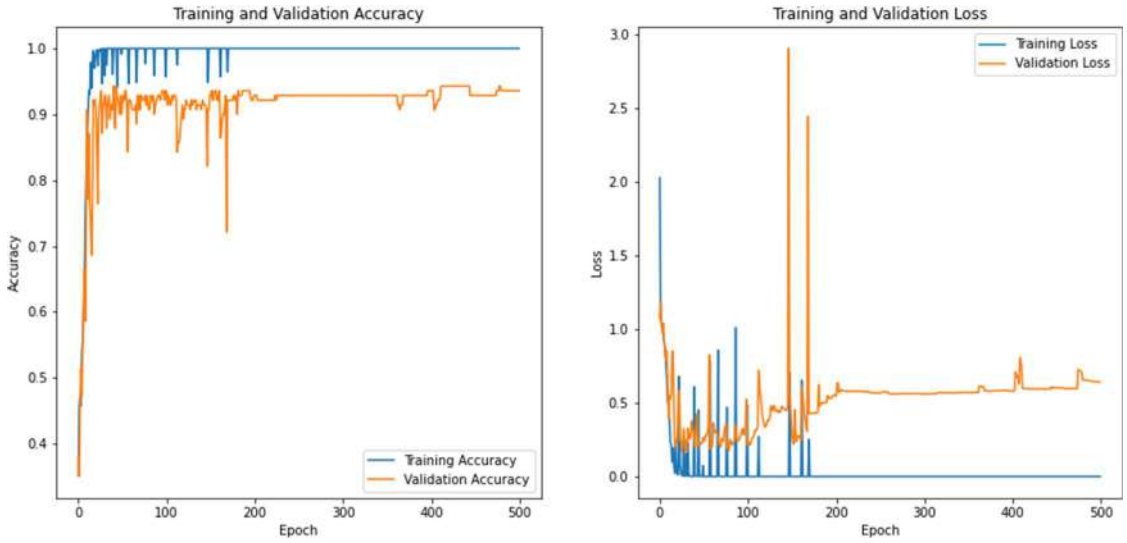
Gambar D.4 menunjukan grafik *loss* dan akurasi pelatihandengan *learning rate* 0,01 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 93,57 % setelah melalui 500 *epochs* dengan *learning rate* 0,01. *Learning rate* adalah parameter *training* untuk menhitung koreksi bobot pada waktu proses *training* Sementara untuk *training loss*mencapai 0,00 dan *validation loss*



Gambar D.4 Grafik Hasil *Training Validation Learning Rate* 0,01

mencapai 0,6413.

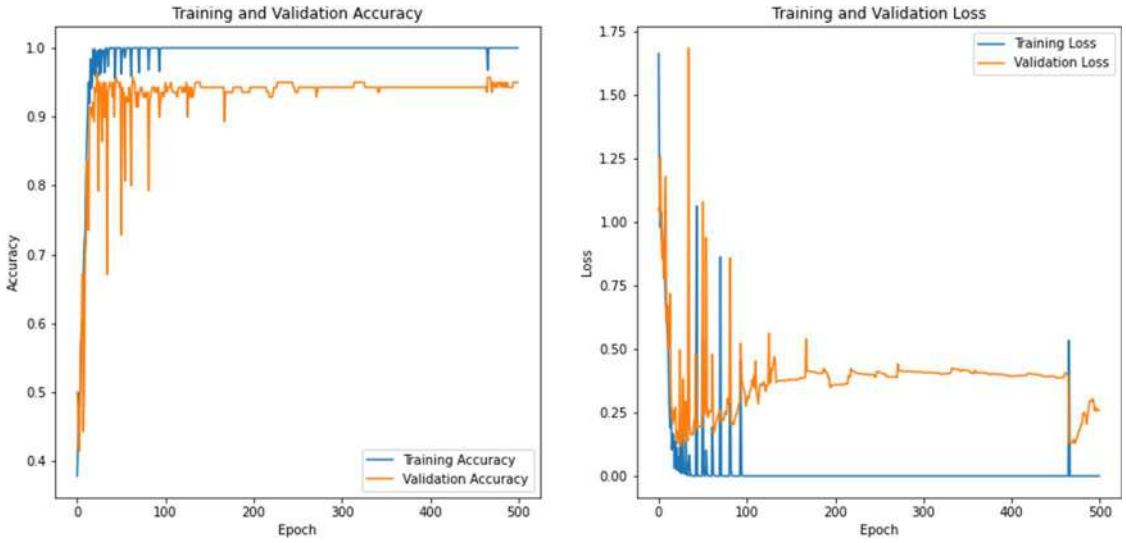
Pada gambar D.5 menunjukan grafik *loss* dan akurasi pelatihan dengan *learning rate* 0,001 dapat dilihat bahwa *training accuracy* mencapai angka 100 % dan *validation accuracy* mencapai 96,43 % setelah melalui 500 *epochs* dengan *learning rate* 0,01. *Learning rate* adalah parameter *training* untuk menhitung koreksi bobot pada waktu proses *training* Sementara untuk *training loss* mencapai 0,00 dan *validation loss* mencapai 0,3636.



Gambar D.5 Grafik Hasil *Training Validation Learning Rate* 0,001

Dari gambar D.6 menunjukan grafik *loss* dan akurasi pelatihan dengan *learning rate* 0,0001 dapat dilihat bahwa *training accuracy* mencapai angka100 % dan *validation accuracy* mencapai 95% setelah melalui 500 *epochs* dengan *learning rate* 0,01. *Learning rate* adalah parameter *training* untuk menghitung koreksi bobot pada waktu proses *training* Sementara untuk *training loss* mencapai0,00 dan *validation loss* mencapai 0,26.

Dari gambar D.5 menunjukan grafik *loss* dan akurasi pelatihan dengan *learning rate*



Gambar D.6 Grafik Hasil *Training Validation Epoch* 0,0001