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.

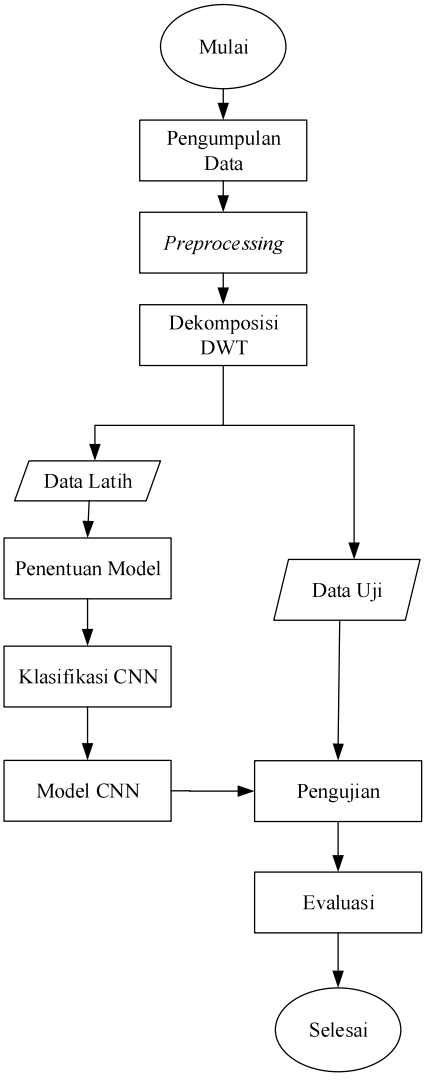
3.2 Instrumen Penelitian

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..

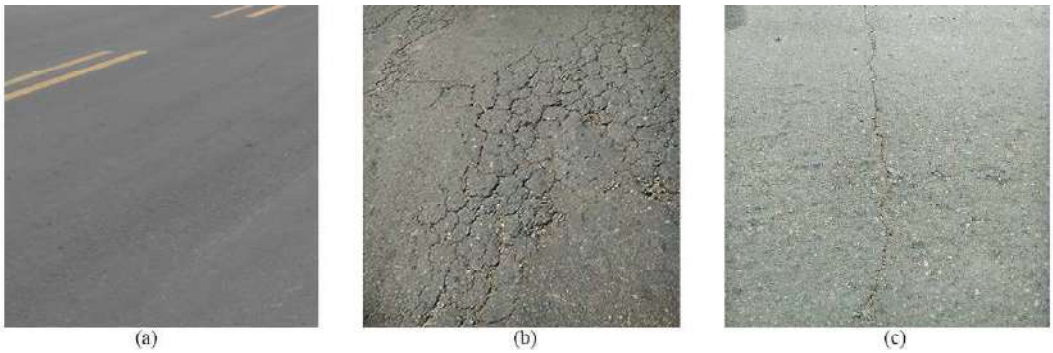


Gambar 3.1 *Flowchart* Penelitian

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.

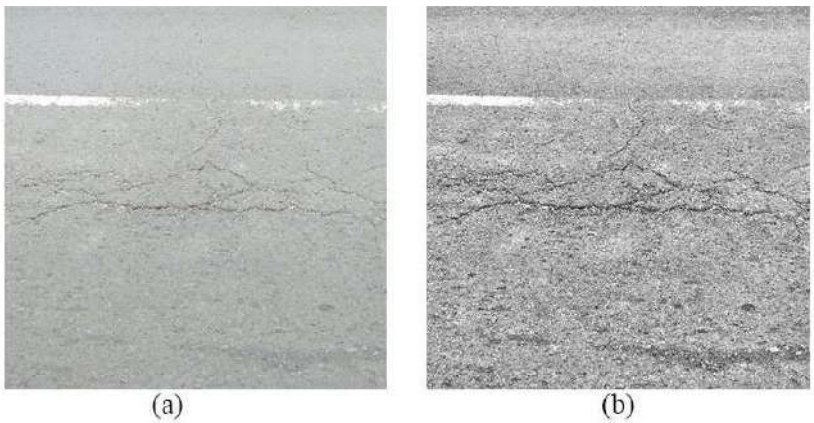
3.5 Preprocessing

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.



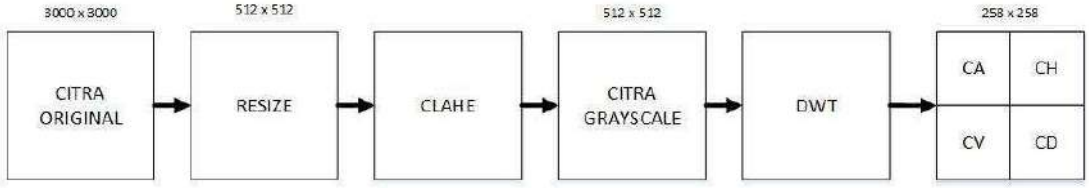
Gambar 3.3 Jenis Citra (a) RGB (b) *Grayscale*

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

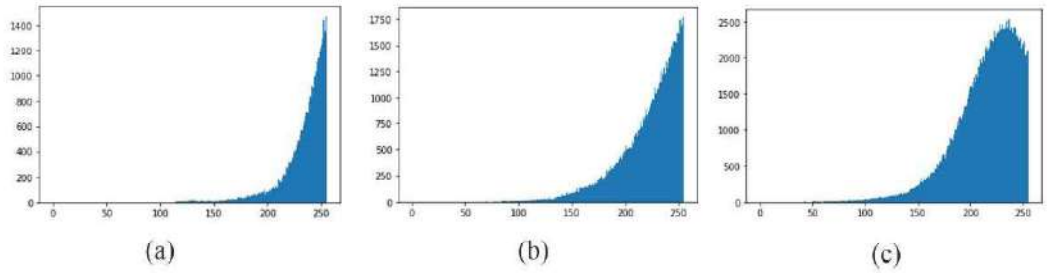
Dekomposisi *discrete Wavelet* transform adalah proses mengurai citra menjadi subgambar (subband) yang memiliki frekuensi dan orientasi berbeda, yang terdiri dari

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.:

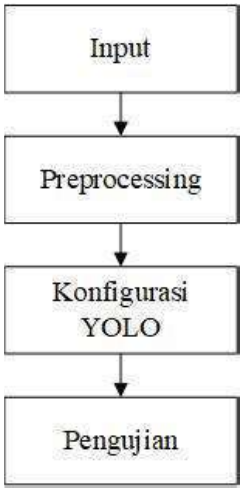
1. Pada langkah pertama *konvolusi*, digunakan kernel berukuran 3x3 dengan total 32 *filter*. *Konvolusi* adalah proses kombinasi antara dua *matrix* yang berbeda untuk menghasilkan *matrix* nilai baru. Setelah tahap *konvolusi* selesai, langkah selanjutnya adalah menerapkan fungsi aktivasi ReLU. Fungsi 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

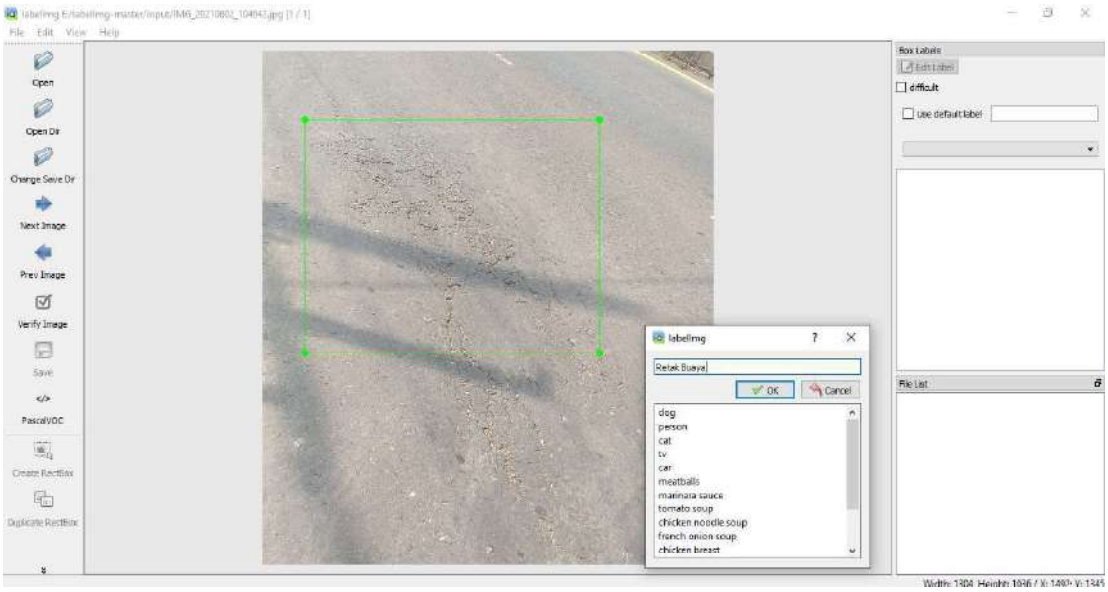


Gambar *3.5 Flowchart* Deteksi Retak Jalan Secara *Real-Time*

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.

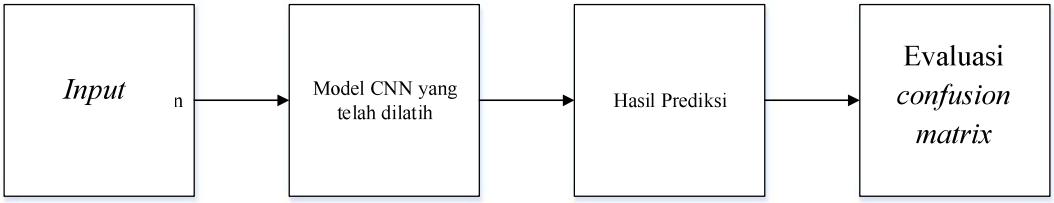
b. Pengujian Model YOLO

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*

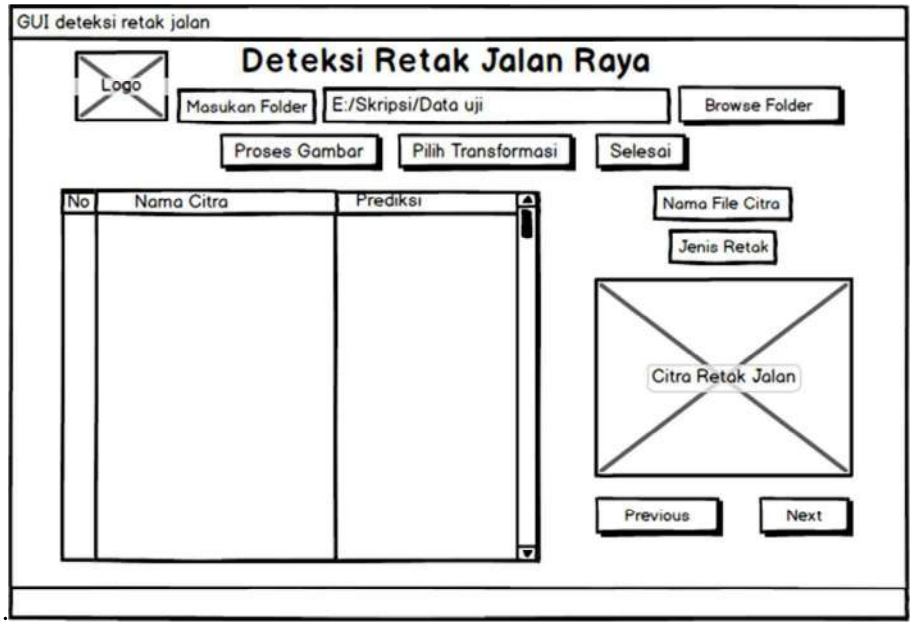
Model CNN yang telah sebelumnya dibuat, kembali menjalani serangkaian pengujian dengan mengubah parameter tertentu. Dalam pengujian ini, parameter yang dimodifikasi adalah nilai *epoch*. Terdapat tiga nilai *epoch* yang digunakan, yakni 100, 300, dan 500. Hasil dari proses pelatihan setiap model menunjukkan 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*.