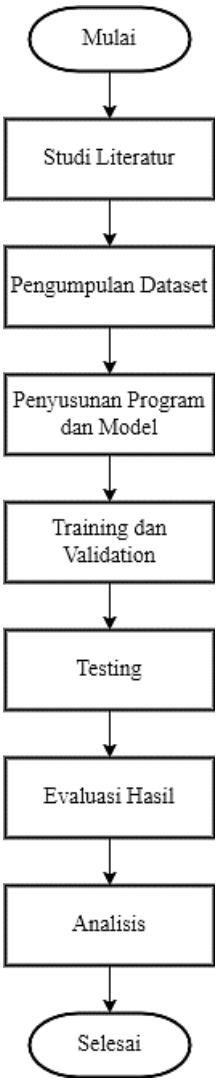
BAB 3
METODE PENELITIAN

Pada Bab 3 ini, peneliti akan mengulas mengenai alat dan bahan yang digunakan untuk mendukung penelitian, termasuk proses pembuatan sistem. Topik yang akan dibahas meliputi alur penelitian, proses pengumpulan dataset, pembuatan dan pengembangan desain sistem deteksi kendaraan, pengembangan model arsitektur, implementasi sistem desain pada pendeteksi kendaraan, serta pengujian sistem untuk mengevaluasi performanya.

3.1 ALUR PENELITIAN

Penelitian ini dimulai dengan melakukan studi literatur, pengumpulan dataset, perancangan sistem, dan dilanjutkan dengan analisis penelitian seperti yang terlihat pada gambar 3.1 di bawah ini :



Gambar 3.1 Rancangan Penelitian

Alur dari penelitian ini dimulai dari studi literatur, dimana peneliti mencari ide atau sumber referensi dalam penelitian. Pada bagian ini peneliti juga menelusuri sumber-sumber tulisan yang pernah dibuat sebelumnya untuk menyelesaikan persoalan yang dihadapi. Selanjutnya yaitu pengumpulan dataset yang dilakukan oleh peneliti dengan cara mengambil dataset secara langsung atau melalui situs web seperti google, kaggle, dan github. Penyusunan program pada penelitian ini menggunakan pemrograman *python* dan google colab yang dibuat menggunakan beberapa *library* yang diperlukan seperti cv2 dan *numpy*. Setelah program selesai maka diperlukan juga penyusunan model. Program dan model selesai, selanjutnya yaitu adanya data *training* yang digunakan untuk melatih dan mengembangkan model dan *validation* yang berfungsi sebagai pembuktian bahwa program yang dibuat sudah tepat sesuai dengan kebutuhan. Tahapan berikutnya merupakan tahap *testing*, dimana model yang telah jadi diuji untuk mengetahui performa algoritma yang sudah dilatih. Tahap *testing* selesai maka selanjutnya yaitu tahap evaluasi hasil, pada tahap ini peneliti dapat melihat hasil program yang dicapai. Hasil dari program tersebut digunakan sebagai landasan untuk mengambil keputusan akhir mengenai langkah selanjutnya, seperti perlu melakukan perbaikan, modifikasi, peningkatan, atau bahkan menghentikan program tersebut. Tahap terakhir yaitu analisis, disini peneliti dapat melihat dan menganalisis performa algoritma yang dipilih dan apa manfaat serta kekurangan dari algoritma atau model tersebut.

3.2 ALAT YANG DIGUNAKAN

Pada sub bab ini membahas mengenai perangkat keras (*hardware*) dan perangkat lunak (*software*) yang akan digunakan pada penelitian ini.

3.2.1 Perangkat Keras (*Hardware)*

Perangkat keras yang digunakan dalam penelitian ini mencakup sebuah laptop yang memiliki spesifikasi sebagai berikut:

1. *Operating System* Windows 11 *Pro* 64-bit
2. *Processor* Intel(R) *Core*(TM) i3-5010U
3. RAM 8,0 GB
4. HDD 465 GB

3.2.2 Perangkat Lunak (*Software)*

Selain perangkat keras, penelitian ini juga menggunakan berbagai perangkat lunak, termasuk:

1. Google *Colaboratory*

   Google colab dipilih sebagai *tools* yang akan digunakan untuk penulisan *source code*. Dipilihnya Google colab karena dapat menjalankan kode Python tanpa perlu melakukan proses instalasi dan setup lainnya. Alasan lain dipilihnya Google colab yaitu *pre-install libraries* yang melimpah, dapat disimpan di *cloud*, dapat berkolaborasi, serta penggunaan GPU (*Graphic Processing Unit*) dan TPU (*Tensor Processing Unit*) gratis untuk proyek *machine learning*. Pada penelitian ini menggunakan GPU sebagai penunjang dalam menjalankan program.

2. Bahasa Pemrograman Python

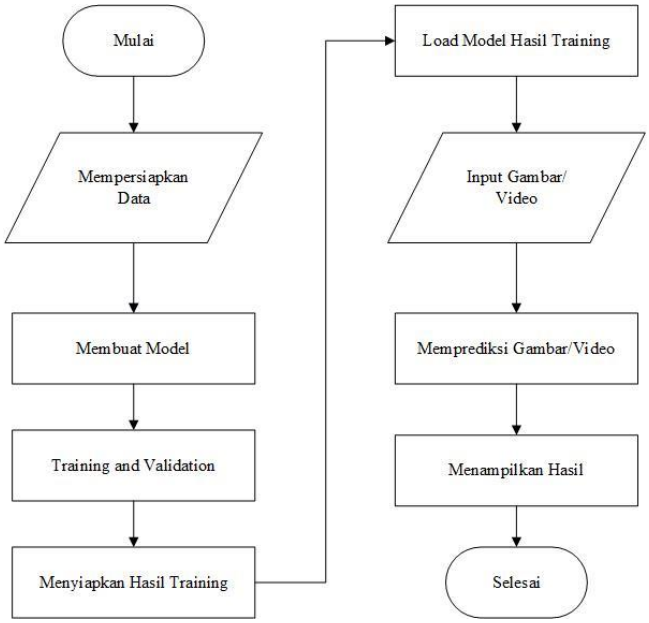
   Bahasa pemrograman yang digunakan untuk pembuatan program pendeteksi kendaraan roda empat dengan metode *Convolutional Neural Network* (CNN) dengan arsitektur U-Net yaitu menggunakan bahasa pemrograman python. Penelitian ini menggunakan python versi 3 dan beberapa *library* yang diperlukan agar program dapat berjalan dengan baik dan benar.

3. Google Chrome versi 108.0.5359.99 (64-bit)

   Google chrome selaku peramban internet digunakan untuk menjalankan Google colab, pada penelitian ini selain menggunakan google chrome juga terhubung dengan google drive dan kaggle. Google drive yang terintegrasi dengan google colab digunakan sebagai penyimpanan *source code* yang nantinya akan dipanggil ketika program dijalankan, dan Kaggle digunakan untuk mengambil dataset.

3.3 RANCANGAN SISTEM

Rancangan sistem pada penelitian ini dibuat menggunakan *flowchart*, dapat dilihat pada gambar 3.2 dibawah ini.



Gambar 3.2 *Flowchart* Sistem

Pada diagram *flowchart* tersebut dapat diuraikan melalui tiga tahapan berikut ini :

1. Tahap Persiapan

   Terdiri dari persiapan perangkat *hardware* dan *software* seperti pada subbab 3.2 diatas. Pada bagian ini termasuk juga persiapan data, mulai dari jumlah data awal hingga data di *pre-processing*.

2. Tahap Perancangan

   Terdiri dari perancangan arsitektur model, melihat akurasi dan *loss* model serta mengekstraknya kedalam tabel csv. Pada bagian ini model juga akan di evaluasi apakah hasilnya sesuai yang diharapkan atau tidak.

3. Tahap Pengujian

   Tahap ini kita akan melihat kinerja model yang dilihat menggunakan *confusion matrix*. Pada tahap ini juga akan melakukan pengujian prediksi model menggunakan beberapa tahapan pengujian.

3.3.1 *Dataset*

Dataset yang digunakan dalam penelitian ini merupakan hasil pengolahan dari KITTI Dataset, yang awalnya dibuat oleh Andreas Geiger, Philip Lenz, dan Raquel Urtasun, sebagaimana tercatat dalam paper berjudul "*Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*." Dataset ini mencakup sebanyak 14.999 gambar kendaraan beserta labelnya. Data tersebut telah diperoleh

dari sumber publik, yakni Kaggle. Kaggle merupakan situs yang menjadi salah satu *platform Data Science* terbesar, digunakan untuk berbagi ide, mendapatkan inspirasi, mempelajari informasi baru dan trik *coding*, serta masih banyak lagi. Pada dataset yang berjumlah 14.999 gambar, dibagi menjadi 49% data *training* yaitu 7481 gambar dan 51% data *testing* yaitu 7518 gambar.

Gambar 3.3 Citra *testing* dataset KITTI

Gambar 3.4 Citra *training* dataset KITTI

3.3.2 Proses *Pre-processing*

1. *Grayscale*

Maksud dari langkah ini adalah untuk menggambarkan gambar dengan tiga lapisan warna, yaitu merah, hijau, dan biru, dalam satu lapisan warna tunggal. Ilustrasi hasil proses perubahan dari skema warna RGB ke skema *Grayscale* dapat dilihat pada gambar 3.5 di bawah ini.

Gambar 3.5 Citra Hasil Konversi ke Bentuk *Grayscale*

2. Normalisasi

Melakukan normalisasi pada piksel citra untuk memastikan bahwa rentang nilai piksel sesuai dengan kebutuhan model. Normalisasi dilakukan dengan membagi nilai piksel yang bernilai 255 agar piksel berada dalam rentang antara 1 dan -1. Normalisasi ini membantu dalam konvergensi model selama pelatihan dan dapat meningkatkan performa model. Untuk normalisasi dapat dilihat pada hasil yang terdapata di Lampiran 1, dimana peneliti membatasi nilai diantara 1 dan -1.

3. Konversi ke Format *Tensors*

Mengonversi citra menjadi format *tensors* yang kompatibel dengan model U-Net CNN. *Tensors* adalah representasi data multidimensi yang digunakan oleh kebanyakan *framework deep learning*, termasuk TensorFlow dan Keras. Peneliti dapat menggunakan fungsi dari *library* seperti NumPy atau TensorFlow untuk mengonversi citra menjadi *tensors*.

```
!pip install tensorflow keras
```

Gambar 3.6 *Install library* tensorflow keras

```
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.optimizers import Adam
```

Gambar 3.7 *Library* untuk Konversi ke Format Tensor

4. Augmentasi Data

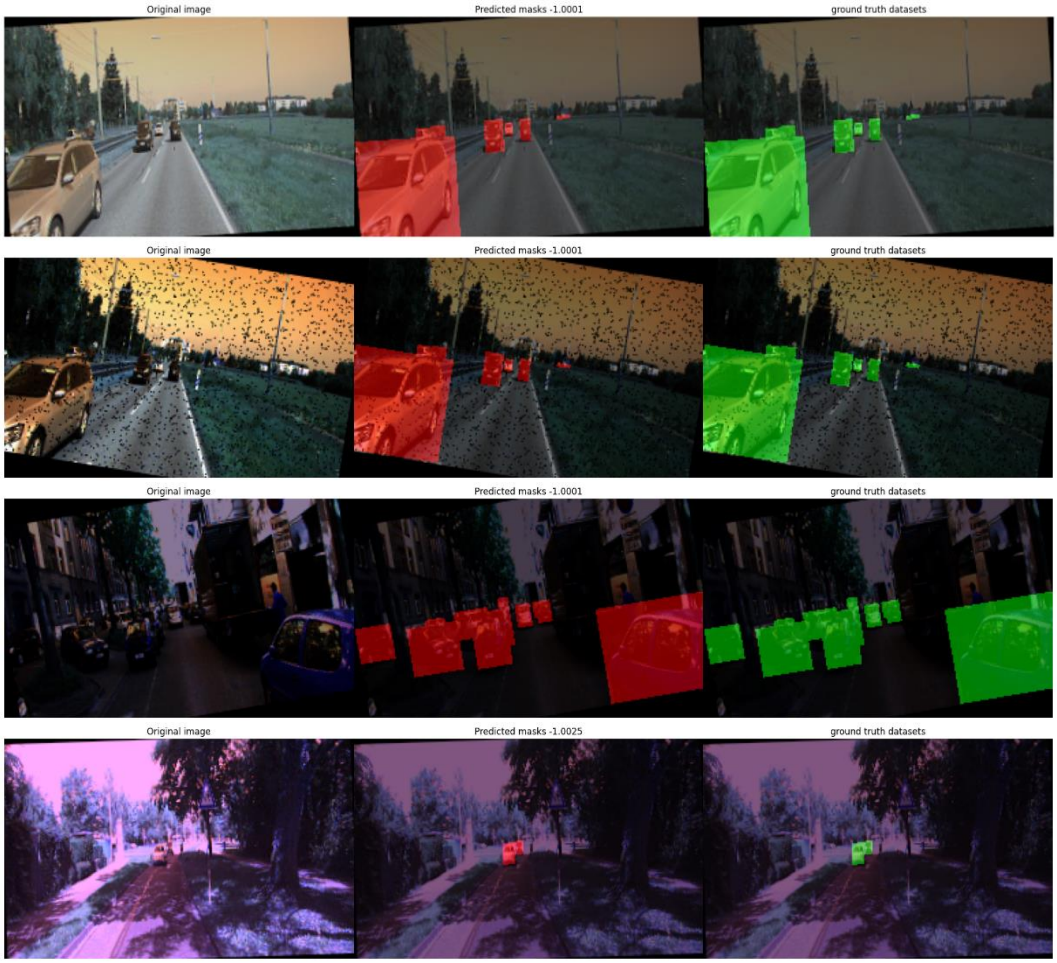
Peneliti dapat menggunakan teknik augmentasi data untuk meningkatkan variasi dataset dan mencegah *overfitting*. Augmentasi data dapat mencakup operasi seperti rotasi, pergeseran, pemotongan, perubahan kecerahan, atau operasi lainnya untuk menghasilkan variasi citra yang baru. Untuk Teknik augmentasi dapat menggunakan *library* seperti “imgaug” atau “Augmentor” untuk menerapkan augmentasi data pada citra.

```
import imgaug.augmenters as iaa
from imgaug.augmentables.segmaps import SegmentationMapsOnImage
```

Gambar 3.8 *Library* untuk Augmentasi Data

```
image_generator = KittiDataGenerator(df,
                                     shape = (IMAGE_WIDTH, IMAGE_HEIGHT),
                                     augmentation = seq)
```

Gambar 3.9 *Script* untuk Augmentasi Data



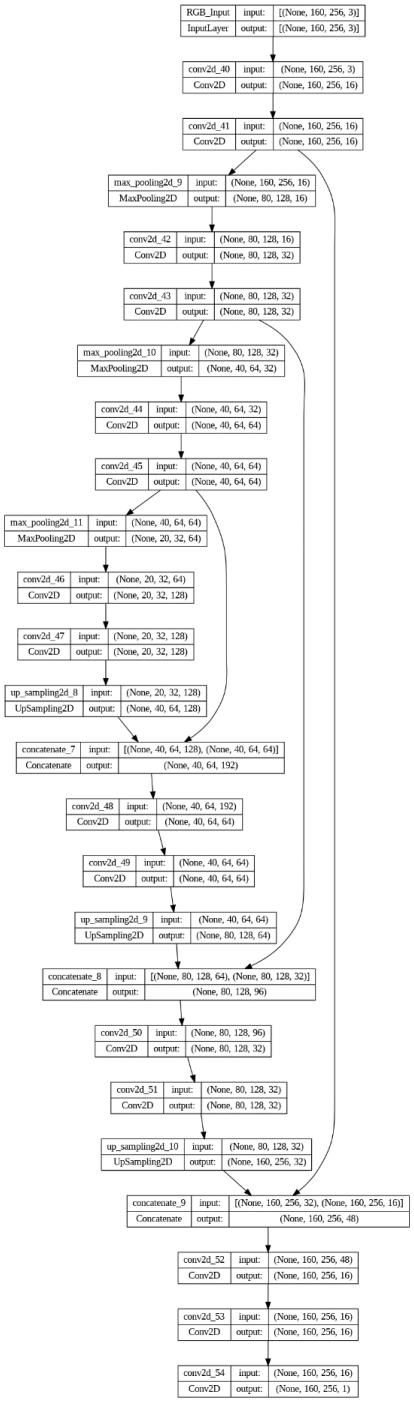
Gambar 3.10 Beberapa contoh hasil dari Augmentasi Data

3.3.3 Arsitektur *Convolutional Neural Network (CNN)*

Pada subbab ini akan dijelaskan bentuk arsitektur CNN menggunakan skenario pembagian dataset yaitu 49% data latih dan 51% data uji. Arsitektur model pada penelitian ini menggunakan arsitektur U-Net untuk model segmentasi citra.

U-Net memiliki struktur yaitu bagian kontraksi (*downsampling*) dan bagian ekspansi (*upsampling*). Bagian kontraksi terdiri dari serangkaian lapisan konvolusi dan *max pooling* yang ditujukan untuk mengekstraksi fitur dari gambar (citra). Bagian ekspansi kemudian menggabungkan fitur yang diekstraksi dengan lapisan konvolusional terbalik dan operasi upsampling untuk membuat peta segmentasi dengan resolusi yang sama dengan citra masukan. Arsitektur U-Net biasanya terdiri dari blok konvolusi berulang, setiap blok terdiri dari dua atau lebih lapisan konvolusi. Dalam skrip yang diberikan, blok konvolusi terdiri dari dua lapisan konvolusi dengan aktivasi ReLU dan *padding* “same”.

Selain itu, juga terdapat operasi penggabungan (*concatenation*) yang menggabungkan fitur-fitur dari bagian kontraksi dengan lapisan konvolusi terbalik (*upsampling*) pada bagian ekspansi. Penggabungan ini membantu menggabungkan data dari berbagai tingkat resolusi dan menyimpan informasi kontekstual penting dalam proses segmentasi. Total parameter model dari penelitian ini adalah 487,297, dengan *learning rate* sebesar 0,0001.



Gambar 3.11 Arsitektur CNN

Berikut adalah penjelasan untuk setiap *layer* (tipe) pada arsitektur U-Net:

1. RGB_*Input* (*InputLayer*)

   Lapisan masukan pertama dari gambar berwarna dengan ukuran (None, 160, 256, 3), di mana "None" adalah ukuran *batch*, 160 adalah tinggi gambar, 256 adalah lebar gambar, dan 3 adalah jumlah saluran warna (RGB: Merah, Hijau, Biru).

2. conv2d_40 (Conv2D)
   - Lapisan konvolusi dengan 16 filter ukuran 3x3.
   - *Output*: (None, 160, 256, 16)
   - Parameter = 448 (jumlah parameter yang dapat dilatih)
   - Terhubung dengan: RGB_*Input*

3. conv2d_41 (Conv2D)
   - Lapisan konvolusi dengan 16 filter ukuran 3x3.
   - *Output*: (None, 160, 256, 16)
   - Parameter = 2,320
   - Terhubung dengan: conv2d_40

4. max_pooling2d_9 (MaxPooling2D)
   - Lapisan pemenggalan spasial dengan faktor 2x2.
   - *Output*: (None, 80, 128, 16)
   - Terhubung dengan: conv2d_41

5. conv2d_42 (Conv2D)
   - Lapisan konvolusi dengan 32 filter ukuran 3x3.
   - *Output*: (None, 80, 128, 32)
   - Parameter = 4,640
   - Terhubung dengan: max_pooling2d_9

6. conv2d_43 (Conv2D)
   - Lapisan konvolusi dengan 32 filter ukuran 3x3.
   - *Output*: (None, 80, 128, 32)
   - Parameter = 9,248
   - Terhubung dengan: conv2d_42

7. max_pooling2d_10 (MaxPooling2D)
   - Lapisan pemenggalan spasial dengan faktor 2x2.

- *Output*: (None, 40, 64, 32)
- Terhubung dengan: conv2d_43

8. conv2d_44 (Conv2D)
   - Lapisan konvolusi dengan 64 filter ukuran 3x3.
   - *Output*: (None, 40, 64, 64)
   - Parameter = 18,496
   - Terhubung dengan: max_pooling2d_10
9. conv2d_45 (Conv2D)
   - Lapisan konvolusi dengan 64 filter ukuran 3x3.
   - *Output*: (None, 40, 64, 64)
   - Parameter = 36,928
   - Terhubung dengan: conv2d_44
10. max_pooling2d_11 (MaxPooling2D)
    - Lapisan pemenggalan spasial dengan faktor 2x2.
    - *Output*: (None, 20, 32, 64)
    - Terhubung dengan: conv2d_45
11. conv2d_46 (Conv2D)
    - Lapisan konvolusi dengan 128 filter ukuran 3x3.
    - *Output*: (None, 20, 32, 128)
    - Parameter = 73,856
    - Terhubung dengan: max_pooling2d_11
12. conv2d_47 (Conv2D)
    - Lapisan konvolusi dengan 128 filter ukuran 3x3.
    - *Output*: (None, 20, 32, 128)
    - Parameter = 147,584
    - Terhubung dengan: conv2d_46
13. up_sampling2d_8 (UpSampling2D)
    - Lapisan *up-sampling* dengan faktor 2x2.
    - *Output*: (None, 40, 64, 128)
    - Terhubung dengan: conv2d_47
14. oncatenate_7 (*Concatenate*)

- Lapisan penggabungan (concatenation) yang menggabungkan hasil dari *up-sampling* dan lapisan conv2d_45.
- *Output*: (None, 40, 64, 192)
- Terhubung dengan: up_sampling2d_8, conv2d_45

15. conv2d_48 (Conv2D)
    - Lapisan konvolusi dengan 64 filter ukuran 3x3.
    - *Output*: (None, 40, 64, 64)
    - Parameter = 110,656
    - Terhubung dengan: *Concatenate*_7
16. conv2d_49 (Conv2D)
    - Lapisan konvolusi dengan 64 filter ukuran 3x3.
    - *Output*: (None, 40, 64, 64)
    - Parameter = 36,928
    - Terhubung dengan: conv2d_48
17. up_sampling2d_9 (UpSampling2D)
    - Lapisan *up-sampling* dengan faktor 2x2.
    - *Output*: (None, 80, 128, 64)
    - Terhubung dengan: conv2d_49
18. *Concatenate*_8 (*Concatenate*)
    - Lapisan penggabungan (concatenation) yang menggabungkan hasil dari *up-sampling* dan lapisan conv2d_43.
    - *Output*: (None, 80, 128, 96)
    - Terhubung dengan: up_sampling2d_9, conv2d_43
19. conv2d_50 (Conv2D)
    - Lapisan konvolusi dengan 32 filter ukuran 3x3.
    - *Output*: (None, 80, 128, 32)
    - Parameter = 27,680
    - Terhubung dengan: *Concatenate*_8
20. conv2d_51 (Conv2D)
    - Lapisan konvolusi dengan 32 filter ukuran 3x3.
    - *Output*: (None, 80, 128, 32)
    - Parameter = 9,248

- Terhubung dengan: conv2d_50

21. up_sampling2d_10 (UpSampling2D)
    - Lapisan *up-sampling* dengan faktor 2x2.
    - *Output*: (None, 160, 256, 32)
    - Terhubung dengan: conv2d_51
22. *Concatenate*_9 (*Concatenate*)
    - Lapisan penggabungan (concatenation) yang menggabungkan hasil dari *up-sampling* dan lapisan conv2d_41.
    - *Output*: (None, 160, 256, 48)
    - Terhubung dengan: up_sampling2d_10, conv2d_41
23. conv2d_52 (Conv2D)
    - Lapisan konvolusi dengan 16 filter ukuran 3x3.
    - *Output*: (None, 160, 256, 16)
    - Parameter = 6,928
    - Terhubung dengan: *Concatenate*_9
24. conv2d_53 (Conv2D)
    - Lapisan konvolusi dengan 16 filter ukuran 3x3.
    - *Output*: (None, 160, 256, 16)
    - Parameter = 2,320
    - Terhubung dengan: conv2d_52
25. conv2d_54 (Conv2D)
    - Lapisan konvolusi dengan 1 filter ukuran 3x3.
    - *Output*: (None, 160, 256, 1)
    - Parameter = 17

    Terhubung dengan: conv2d_53

Dalam keseluruhan arsitektur ini, jalur konvolusi dan jalur penggabungan (*concatenation*) digunakan untuk melakukan segmentasi gambar. Jaringan ini berusaha untuk mengidentifikasi dan memisahkan objek-objek dalam gambar dengan mengambil informasi fitur pada berbagai tingkat skala dan mendeteksi pola yang lebih kompleks seiring dengan kedalaman lapisan.

3.3.4 Rancangan Pengujian

Penelitian ini melakukan evaluasi terhadap model yang dihasilkan oleh *Convolutional Neural Network* (CNN). Evaluasi dilakukan dalam dua tahap, yaitu tahap *training* dan *testing*. Tahap *training* melibatkan pengujian model CNN menggunakan data latih yang terdiri dari 7481 gambar, yang terbagi menjadi data *train* dan *validation*.

Pada tahap *testing*, model yang telah melalui tahap pelatihan diuji menggunakan 7518 gambar uji yang berbeda. Tujuan dari pengujian ini adalah untuk menguji apakah model dapat mengklasifikasikan gambar dengan baik, sehingga dapat menilai performa model yang dihasilkan.

Dengan kata lain, penelitian ini dilakukan untuk mengevaluasi model CNN yang telah dilatih menggunakan dua tahap, yaitu tahap pelatihan dengan data latih berjumlah 7481 gambar dan tahap pengujian dengan 7518 gambar uji yang berbeda untuk menguji kemampuan model dalam mengklasifikasikan gambar dengan baik.

3.3.5 Pelatihan Model

Pelatihan model (*training*) untuk melatih algoritma CNN dalam mengenali datasetnya dan membentuk sebuah model berdasarkan pelatihan yang akan dijelaskan pada subbab ini, berisi tentang penentuan parameter yang dibutuhkan didalam model CNN. Bagian ini juga menunjukkan program untuk arsitektur dari model CNN, dimana akan ditunjukkan proses konvolusi dengan aktivasi ReLU dan *pooling*. Selanjutnya proses augmentasi data atau *preprocessing* data, mulai dari menentukan *batch size* dan nilai *epoch*. Setelah semua selesai dilanjutkan dengan melihat grafik model dan *save model.*

3.3.6 Model Hasil *Training*

Model hasil *training* ini merupakan model yang akan digunakan untuk membuat prediksi pada data baru. Model ini berekstensi (.h5) dimana dalam model ini terdapat arsitektur model, bobot (*weights*), serta konfigurasi lain yang diperlukan untuk merekonstruksi model yang telah dilatih.

```
[40] from tensorflow.keras.models import load_model

     # Save the model
     model.save('kitti_segmentation_model.h5')
```

Gambar 3.12 *Source Code Save Model*

3.4 TAHAP PENGUJIAN

Pada tahap pengujian, sistem yang telah dirancang akan menghasilkan model hasil *training*. Tujuan dari tahap ini adalah untuk menguji kemampuan sistem dan memastikan apakah sistem berfungsi dengan baik sesuai dengan kebutuhan dan harapan peneliti. Penelitian ini terdiri dari beberapa pengujian yang dilakukan untuk mencapai tujuan tersebut, yaitu:

3.4.1 Pengujian Deteksi pada Citra dengan Subjek Kendaraan dari Posisi Depan

Pada skenario pengujian ini peneliti akan menggunakan sampel data citra yang diambil menggunakan kamera pada subjek kendaraan. Skenario pengujian ini dilakukan dengan cara memprediksi subjek yang berada di jalan dengan posisi kamera didepan kendaraan. Pengujian ini akan memperlihatkan apakah sistem yang telah dibuat dapat menguji kendaraan dari posisi depan.

3.4.2 Pengujian Deteksi pada Citra dengan Subjek Kendaraan dari Posisi Belakang

Pada skenario pengujian ini peneliti akan menggunakan sampel data citra yang diambil menggunakan kamera dalam bentuk foto berupa subjek kendaraan di jalan. Skenario pengujian ini dilakukan dengan cara memprediksi subjek yang berada di jalan dengan posisi kamera didepan kendaraan. Pengujian ini akan memperlihatkan apakah sistem yang telah dibuat dapat menguji kendaraan dari posisi depan ketika dalam kondisi ramai dan kendaraan tidak terlihat dengan sempurna seluruh badannya.

3.4.3 Pengujian Deteksi pada Citra dengan Subjek Kendaraan dari Posisi Depan dan Belakang

Pada skenario pengujian ini peneliti akan menggunakan sampel data citra yang diambil menggunakan kamera dalam bentuk foto berupa subjek kendaraan di jalan. Skenario pengujian ini dilakukan dengan cara memprediksi subjek yang berada di jalan dengan posisi kamera didepan dan dibelakang kendaraan. Pengujian ini akan memperlihatkan apakah sistem yang telah dibuat dapat menguji kendaraan dari posisi depan ketika dalam kondisi ramai dan kendaraan tidak terlihat dengan sempurna seluruh badannya.

3.4.4 Pengujian IoU untuk Melihat Performa Model

Pada skenario pengujian terakhir, peneliti akan melihat performa model menggunakan metriks pengujian IoU. Pengujian ini akan melihat apakah model yang telah dibuat mengalami *Overfitting*, *Underfitting* atau *Good Fit*. Dimana hasil dari pengujian ini bergantung pada nilai IoU dan *Loss* pada pelatihan model.