Smart Comp: Jurnalnya Orang Pintar Komputer
Volume 14, Nomor 1, Januari 2025, hlm. 218-229
Terakreditasi Sinta, Peringkat 4, SK No. 105/E/KPT/2022
DOI : 10.30591/smartcomp.v13i1.8286

P-ISSN : 2089-676X
E-ISSN : 2549-0796

Deteksi Ketersediaan Lahan Parkir Mobil Menggunakan Yolo V4 Berbasis Website

I Made Bayu Suarjaya[1], Giri Wahyu Wiriasto[2], Paniran[3]
[1,2,3)]Program Studi Teknik Elektro, Fakultas Teknik, Universitas Mataram
Email: [1]imadebayusuarjaya@gmail.com, [2]giriwahyuwiriasto@gmail.com, [3]paniranmt@yahoo.com

(Naskah masuk: *15 Januari 2025*, diterima untuk diterbitkan: *20 Januari 2025*)

Abstrak: *Sistem deteksi ketersediaan lahan parkir mobil merupakan solusi inovatif untuk mengatasi masalah parkir yang sering kali penuh dan sulit dipantau secara manual. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem deteksi ketersediaan lahan parkir mobil menggunakan algoritma YOLOv4 yang terintegrasi dengan platform berbasis website. Sistem ini bekerja dengan mendeteksi mobil yang terparkir pada suatu area parkir menggunakan teknik deteksi objek berbasis deep learning, yaitu YOLOv4. Pengguna dapat mengunggah gambar atau video area parkir melalui website, yang kemudian diproses oleh model YOLOv4 untuk mendeteksi kendaraan yang terparkir. Sistem ini tidak hanya menghitung jumlah kendaraan yang ada, tetapi juga memberikan informasi tentang ketersediaan lahan parkir. Lahan kosong akan ditandai dengan warna hijau, sedangkan lahan yang terisi akan ditandai dengan warna merah pada tampilan antarmuka pengguna. Data training yang digunakan dalam pengembangan model ini sebanyak 130, sementara data uji sebanyak 33. Dengan parameter tersebut, model YOLOv4 berhasil mencapai Mean Average Precision (mAP) sebesar 100%, yang menunjukkan tingkat akurasi deteksi yang sangat tinggi. Dengan solusi ini, diharapkan pengelolaan area parkir menjadi lebih efisien dan pengguna dapat dengan mudah mengetahui ketersediaan lahan parkir tanpa kesulitan.*

Kata Kunci – *Deteksi Parkir; Yolov4, Sistem Berbasis Website; Ketersediaan Lahan Parkir;Deep Learning*

Detection Of Car Parking Space Availability Using Website-Based Yolo V4

Abstract: *The car park availability detection system addresses the challenges of monitoring and managing parking lots that are often overfilled. This research focuses on designing a detection system using the YOLOv4 algorithm, integrated with a web-based platform. The system utilizes deep learning-based object detection to identify vehicles in parking areas. Users can upload images or videos of the parking lot through the website, which are processed by the YOLOv4 model to detect parked cars and determine parking space availability. The interface highlights empty spaces in green and occupied ones in red, making it easier for users to identify available spots. The model was developed using 130 training data samples and tested with 33 data samples. It achieved a remarkable Mean Average Precision (mAP) of 100%, demonstrating its high detection accuracy. This innovative solution is expected to enhance parking management efficiency, enabling users to access real-time information about parking availability conveniently and effectively.*

Keywords – *Parking Detection; Yolov4; Web-based System; Parking Lot Availability; Deep Learning*

1. PENDAHULUAN

Saat ini, jumlah kendaraan roda dua atau lebih semakin meningkat, terutama di kota-kota besar. Jumlah kendaraan di Indonesia terus meningkat setiap tahunnya, baik mobil penumpang, sepeda motor, bus, dan barang. Data dari BPS (Badan Pusat Statistik) menunjukkan bahwa jumlah kendaraan penumpang di Indonesia pada tahun 2020 mencapai 15.797.451, naik dari 14.830.698 pada tahun 2018 dan 15.592.419 pada tahun 2019[3].

Jumlah kendaraan roda dua atau lebih yang memarkir kendaraannya di tempat yang tidak sesuai disebabkan oleh kebutuhan parkir yang meningkat seiring dengan peningkatan jumlah kendaraan di Indonesia[5]. Selain itu, masalah ini sering terjadi karena tidak adanya informasi tentang jumlah ruang parkir yang tersedia di tempat-tempat tertentu, seperti pusat perbelanjaan,

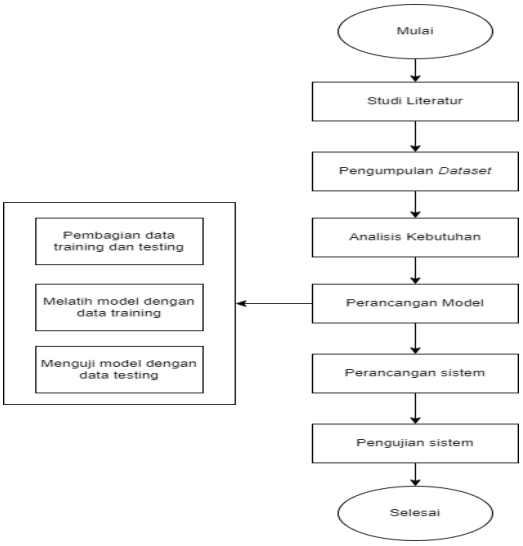
kantor, tempat wisata, dan lain-lain. Akibatnya, pengendara seringkali harus berkeliling untuk mencari tempat parkir yang masih kosong, tetapi jika tidak ditemukan tempat parkir yang tersedia, mereka memaksakan untuk parkir sembarangan di tempat yang tidak tepat.

Perkembangan teknologi, khususnya di bidang komputer, telah memberi manusia banyak kemudahan. Hampir setiap kantor, perusahaan, industri, rumah tangga, atau sekolah menggunakan komputer sebagai alat bantu untuk berbagai macam kebutuhan. Pengelolaan lahan parkir adalah salah satu cara teknologi komputer membantu aktivitas manusia. Sekarang ada sistem manajemen parkir yang terkomputerisasi yang membuatnya lebih mudah bagi pengunjung untuk menemukan tempat parkir kosong[11]. Sistem manajemen parkir adalah program yang dapat mengoptimalkan penggunaan lahan parkir. Saat ini, teknologi tempat parkir telah muncul untuk menggantikan metode konvensional yang hanya menunjukkan jumlah tempat kosong atau penuh pada papan di depan pintu parkir.

Berdasarkan permasalahan yang telah diuraikan di atas, maka pada penelitian ini akan menerapkan deteksi ketersediaan lahan parkir mobil menggunakan algoritma yolov4. Penelitian ini diharapkan dapat mempermudah masyarakat untuk mengetahui kapasitas parkir mobil di suatu tempat. Penelitian ini diharapkan dapat membantu masyarakat mengetahui kapasitas parkir mobil di suatu tempat dengan menggunakan algoritma Yolov4 untuk mendeteksi ketersediaan lahan parkir berdasarkan masalah yang telah diuraikan di atas.

2. METODE PENELITIAN

Penelitian ini dilakukan dari awal hingga akhir sesuai dengan alur penelitian yang digambarkan dalam Gambar 1. Alur penelitian digunakan oleh penulis dalam pelaksanaan penelitian ini agar hasil yang dicapai tidak menyimpang dari tujuan yang telah ditetapkan sebelumnya. Alur penelitian dapat dilihat dalam Gambar 1 berikut.



Gambar 1. Alur Penelitian

2.1. Studi Literatur

Studi literatur dibutuhkan sebagai penunjang dalam menyelesaikan penelitian, yaitu dengan mengumpulkan informasi-informasi yang berkaian dengan topik penelitian. Pada studi literatur ini berkaitan dengan klasifikasi sampah yang bersumber dari buku-buku, jurnal, daln lain-lain sebagai referensi.

2.2. Pengumpulan Dataset

Data yang digunakan dalam penelitian ini berupa data mobil yang ada di parkiran yang terdiri dari 1 class yaitu class mobil. Jumlah data yang digunakan sebesar 163 data, 130 sebagai data *training* dan 33 sebagai data uji.

2.3. Analisis Kebutuhan

Adapun berbagai kebutuhan yang mendukung untuk dilakukannya penelitian ini mencakup hardware dan software yang dimana akan dirincikan sebagai berikut:

Hardware:

1. 1 buah Laptop
2. 1 buah Bardi Smart IP Camera IP67
3. Software:
4. Sistem operasi windows 11.
5. Microsoft Word untuk membuat laporan penelitian.
6. Aplikasi Draw.io untuk membuat diagram
7. Visual Studio Code untuk kode editor dalam melakukan perancangan model dan perancangan sistem menggunakan flask.
8. Python sebagai bahasa pemrograman dalam machine learning.

2.4. Perancangan Model

Perancangan model dilakukan dengan tahapan awal dengan melakukan pembagian dataset akan dilakukan dengan pembagian untuk data training 80% dan data testing 20%. Tahap selanjutnya adalah melatih model YOLOv4 dengan data training yang sudah ada dan akan dilakukan pengujian model dengan menggunakan data latih.

2.5. Perancangan Sistem

Pembuatan aplikasi pada penelitian ini berbasis website dengan menggunakan framework Flask untuk mendeploy model dengan menggunakan opencv sehingga framework flask dapat berjalan menggunakan yolo.

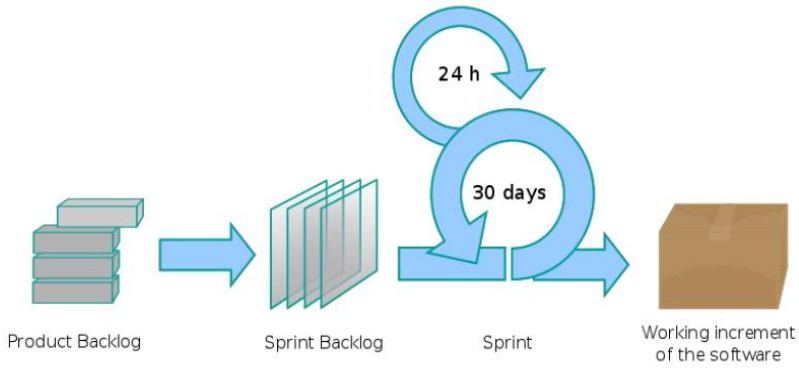
2.6. Pengujian Sistem

Pengujian sistem mengevaluasi performa daripada sistem training dan sistem testing yang telah dibuat. Pengujian dilakukan untuk memastikan sistem dapat berjalan dengan baik.

2.7. Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak yang digunakan pada penelitian ini adalah metode Agile Software Development yaitu scrum. Scrum merupakan sebuah kerangka kerja untuk menyelesaikan pekerjaan-pekerjaan yang kompleks dan selalu berubah. Kerangka Scrum digunakan untuk menjawab persoalan adaptif yang kompleks, menghasilkan kreatifitas dan inovasi. Sprint merupakan inti dari metode Scrum yang merupakan batasan waktu yang dalam 1 bulan atau kurang dimana sebuah ikremen yang selesai, berfungsi dan berpotensi untuk dikembangkan. Proses sprint biasanya memiliki durasi waktu yang konsisten. Jika proses sprint tahap pertama sudah selesai maka dilanjutkan dengan proses sprint selanjutnya. Tahapan Scrum ini terdiri dari product log, sprint backlog, sprint, working increment of the software (Prabowo &

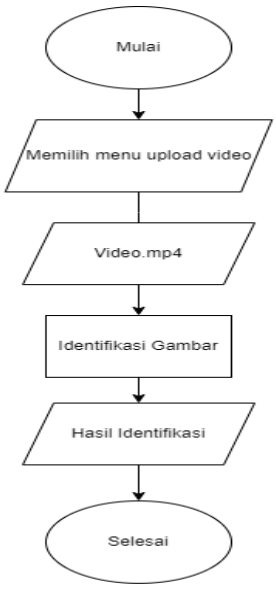
Wiguna, 2021). Tahapan-tahapan penelitian dalam metode agile scrum ini dibagi menjadi beberapa tahapan, yang dapat dilihat pada gambar 2.



Gambar 2. Metode Pengembangan Perangkat Lunak

3. HASIL DAN PEMBAHASAN

3.1. Proses Training Model

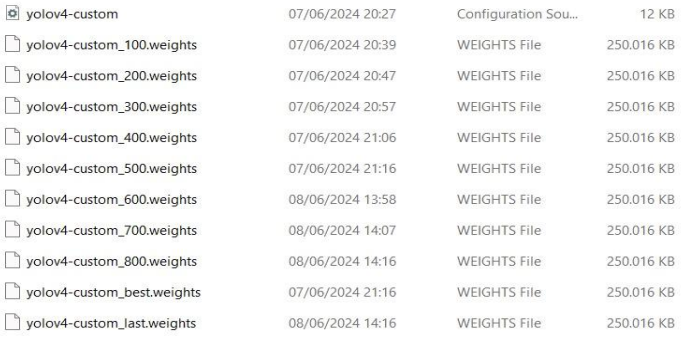


Gambar 3. Proses *Training* Model

Proses pelatihan YOLOv4 dapat dijelaskan dengan menggunakan diagram alir yang terdapat pada Gambar 7 dengan ada lapisan input, backbone, neck dan juga Dense Prediction. Proses pelatihanya sendiri dapat dijelaskan sebagai berikut :

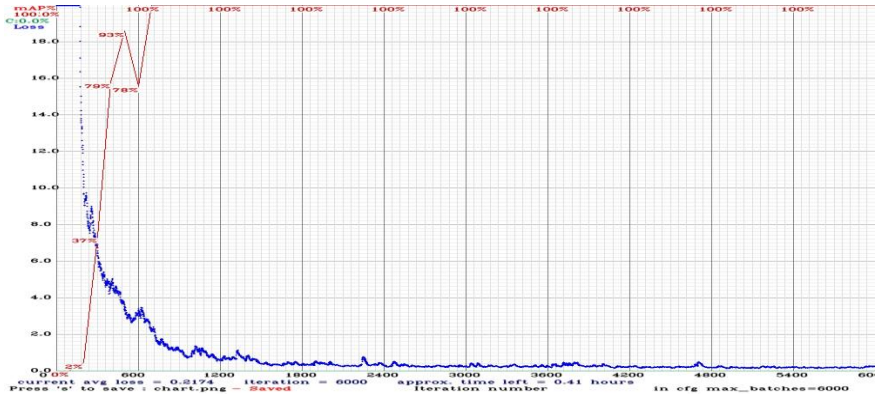
1. Awal awal citra yang masuk ke dalam blok input akan di resize sesuai resolusi yang ada pada lapisan input. Resolusi yang digunakan di model yang dibangun adalah menggunakan resolusi 224 × 224. Penggunaan resolusi input dapat diubah dengan ketentuan dapat dibagi dengan nilai 32. Citra yang masuk pada lapisan input ini secara default yang diterima adalah citra dengan 3 kanal yaitu RGB.
2. Citra yang telah melalui blok input akan melalui backbone yang akan melakukan pengekstran ciri yang ada pada citra. Backbone yang digunakan sendiri adalah CSP-Darknet-53. Proses yang terjadi pada citra adalah citra akan melalui proses konvolusi pada Darknet-53, proses konvolusi ini berguna untuk dapat mengekstraksi fitur fitur yang hierarki di citra tersebut. Lapisan konvolusi yang ada akan melakukan deteksi terhadap pola-pola yang ada diberbagai tingkat resolusi.
3. Hasil konvolusi yang dari backbone akan masuk ke dalam neck yang akan menyatukan ekstraksi fitur dan memperluas informasi yang diperoleh.
4. Setelah melalui neck maka akan masuk ke dense prediction, dimana terdapat YOLO layer yang akan melakukan klasifikasi dan juga pembuatan bounding box dengan cara membagi data menjadi grid cell dan di setiap cell ada bounding box sampai disetiap bounding box dilakukan prediksi.
5. Hasil prediksi dari data tersebut memngkinkan adanya bounding box yang mengalami overlapping, untuk itu dilakukan tahap NMS atau Non Max Suppression, di mana akan dilakukan perbandingan confidance setiap bounding box dengan groud trouth. Hasil prediksi adalah bounding box yang memiliki confidance terbesar.

Proses training yang dilakukan akan menghasilkan weight atau bobot dari model yang disimpan pada file backup. Bobot yang di simpan sendiri adalah bobot setiap 100 iterasi yang dilakukan, bobot final, bobot terakhir, dan bobot 56 yang paling bagus. Isi folder backup yang meyimpan bobot dapat dilihat pada Gambar 4.



Gambar 4. Bobot di *Folder Backup*

Saat Proses training dilakukan dengan perintah darknet yang menggunakan -map menghasilkan chart yang digunakan untuk mengetahui proses pelatihan yang dilakukan oleh model saat iterasi berjalan dengan treshold 0.25. Chart tersebut dapat dilihat pada Gambar 5.



Gambar 5. Hasil Proses *Training*

Pada gambar 5 dapat di analisa bahwa pada saat proses training dilakukan averange loss yang dihasilkan semakin menurun tiap iterasi yang dijalankan. Averange loss yang diperoleh adalah 0.2174 dengan jumlah iterasi sebesar 6000 iterasi. Sedangkan mAP yang di dapatkan dari hasil training dengan bobot paling bagus adalah 100%, dimana pada setiap iterasi yang dilakukan model berusaha untuk meningkatkan mean averange precision yang di dapatkan meskipun di iterasi 700 sampai 1800 masih ada penurunan mAP. Nilai mAP yang 100% yang didapatkan oleh model sendiri diakibatkan pada proses training yang dilakukan data uji nya memiliki itensitas cahaya yang cukup bagus.

3.2. Implementasi YOLOv4 ke Sistem

Adapun proses proses dalam melakukan implementasi YOLOv4 ke sistem dapat dijelaskan sebagai berikut:

1. Import library yang dibutuhkan

```
from flask import Flask,render_template,request,Response, redirect, url_for, send_file
import cv2 as cv
from realtime
import Realtime
import time import numpy as np
import os import cv2
```

Flask digunakan untuk membuat antarmuka web, termasuk merender halaman HTML (render_template), menangani input pengguna (request), mengarahkan ke URL tertentu (redirect, url_for), dan mengirim file hasil analisis (send_file). Library OpenCV (cv2) digunakan untuk membaca dan memproses gambar atau video, dengan dukungan NumPy untuk manipulasi array. Modul Realtime yang diimpor dari realtime.py kemungkinan berisi logika deteksi objek, seperti YOLO v4, yang relevan dengan proyek deteksi mobil di parkiran. Library os digunakan untuk mengelola file dan direktori.

2. Memulai aplikasi berbasis flask

```
app = Flask(__name__)
```

app = Flask(__name__) digunakan untuk membuat *instance* aplikasi Flask yang menjadi pusat pengaturan dan logika aplikasi *web*. Parameter __name__ digunakan untuk memberi tahu Flask lokasi modul aplikasi, sehingga dapat menemukan *file* terkait seperti *template* dan *file* statis. Objek *app* ini nantinya digunakan untuk mendefinisikan rute, mengatur konfigurasi, dan menjalankan aplikasi *web*.

3. Tempat menyimpan *file* yang diunggah

```
UPLOAD_FOLDER = 'static/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

Kode di atas digunakan untuk mengatur direktori penyimpanan *file* yang diunggah dalam aplikasi Flask. Variabel UPLOAD_FOLDER didefinisikan dengan nilai 'static/', yang menunjuk ke *folder* tempat *file* statis seperti gambar atau dokumen disimpan. Konfigurasi ini diterapkan pada aplikasi melalui app.config['UPLOAD_FOLDER']=UPLOAD_FOLDER, sehingga aplikasi dapat mengetahui lokasi penyimpanan *file* yang diunggah pengguna. Pengaturan ini sering digunakan dalam implementasi fitur *unggah file* pada aplikasi *web*.

4. Sebagai halaman *routing* aplikasi

```
@app.route('/', methods=['GET','POST'])
def halaman():
    return render_template("index.html")
```

Kode di atas digunakan untuk mendefinisikan rute utama aplikasi Flask yang terletak di URL '/', yaitu halaman utama aplikasi. Dengan dekorator @app.route('/', methods=['GET', 'POST']), rute ini dapat menangani permintaan HTTP GET untuk memuat halaman dan POST untuk menerima data dari pengguna. Fungsi halaman() yang didefinisikan di dalamnya akan dijalankan setiap kali rute ini diakses, dan akan merender *file* HTML bernama index.html menggunakan fungsi render_template. Hal ini memungkinkan aplikasi untuk menampilkan antarmuka pengguna berbasis *template* HTML di halaman utama

5. Menggambar *bounding box*

```
def draw_yolo_box(frame, bbox, img_width, img_height, color=(0, 255, 0)):
  x_center, y_center, width, height = bbox
```

Fungsi draw_yolo_box dirancang untuk menggambar kotak pembatas (*bounding box*) pada sebuah *frame* gambar atau video berdasarkan hasil deteksi YOLO. Fungsi ini menerima beberapa parameter: *frame* sebagai gambar atau video tempat kotak akan digambar, bbox yang berisi informasi *bounding box* dalam format [x_center, y_center, *width*, *height*], serta img_*width* dan img_*height* untuk mengonversi koordinat relatif *bounding box* menjadi koordinat absolut. Selain itu, parameter color menentukan warna kotak yang akan digambar, dengan nilai default hijau (0, 255, 0) dalam model warna BGR. Baris x_center, y_center, *width*, *height* = bbox digunakan untuk memisahkan nilai koordinat pusat, lebar, dan tinggi *bounding box* untuk mempermudah penghitungan posisi kotak pada *frame*. Fungsi ini biasanya digunakan dalam kombinasi dengan pustaka seperti OpenCV untuk visualisasi hasil deteksi objek.

6. Menggambar *bounding box* pada sebuah gambar atau video

```
def draw_yolo_box(frame, bbox, img_width, img_height, color=(0, 255, 0)):
    x_center, y_center, width, height = bbox

    # Konversi koordinat relatif ke piksel absolut
    x_center_abs = int(x_center * img_width)
    y_center_abs = int(y_center * img_height)
    width_abs = int(width * img_width)
    height_abs = int(height * img_height)

    # Hitung titik kiri atas dan kanan bawah
    top_left_x = int(x_center_abs - width_abs / 2)
    top_left_y = int(y_center_abs - height_abs / 2)
    bottom_right_x = int(x_center_abs + width_abs / 2)
    bottom_right_y = int(y_center_abs + height_abs / 2)

    # Gambar kotak di frame dengan warna sesuai parameter
```

```
        cv.rectangle(frame,   (top_left_x,   top_left_y),   (bottom_right_x,
bottom_right_y), color, 2)
```

Kode di atas mendefinisikan fungsi draw_yolo_box yang digunakan untuk menggambar *bounding box* (kotak pembatas) pada sebuah *frame* gambar atau video. Fungsi ini menerima parameter berupa *frame* (gambar atau *frame* video), bbox (*bounding box* dalam format relatif: x_center, y_center, *width*, *height*), img_*width* dan img_*height* (dimensi gambar), serta color (warna kotak, dengan nilai *default* hijau). Pertama, koordinat *bounding box* yang bersifat relatif diubah menjadi nilai piksel absolut berdasarkan ukuran gambar. Kemudian, titik kiri atas dan kanan bawah dari kotak dihitung untuk menentukan posisi dan ukuran *bounding box* yang akan digambar. Terakhir, fungsi cv.rectangle digunakan untuk menggambar kotak dengan warna yang ditentukan pada *frame*, menggunakan ketiga koordinat yang telah dihitung dan ketebalan garis 2 piksel. Fungsi ini memungkinkan untuk menampilkan hasil deteksi objek dengan visualisasi *bounding box* yang jelas pada gambar atau video.

7. Menghitung *Intersection over Union* (IoU) antara dua *bounding box*

```
 ef calculate_iou(box1, box2):
    x1, y1, w1, h1 = box1
    x2, y2, w2, h2 = box2


    # Menghitung area overlap
    x11 = max(x1, x2)
    y11 = max(y1, y2)
    x21 = min(x1+w1, x2+w2)
    y21 = min(y1+h1,y2+h2)

    intersection_area = max(0, x21 - x11) * max(0, y21 - y11)

    # Menghitung area total dari dua bounding box
    box1_area = w1 * h1
    box2_area = w2 * h2

    union_area = box1_area + box2_area - intersection_area

    iou = intersection_area / union_area
    # print(iou)
    return iou
```

Fungsi calculate_iou digunakan untuk menghitung metrik *Intersection over Union* (IoU) antara dua *bounding box*, yang sering digunakan dalam deteksi objek untuk mengevaluasi sejauh mana dua kotak pembatas saling tumpang tindih. Fungsi ini menerima dua *bounding box* sebagai *input*, yang masing-masing berformat [x, y, w, h], di mana x dan y adalah koordinat titik kiri atas, dan w dan h adalah lebar dan tinggi kotak. Fungsi ini pertama-tama menghitung area tumpang tindih antara kedua *box* dengan menentukan koordinat titik kiri atas dan kanan bawah dari *area overlap*, kemudian menghitung luas *area overlap*. Selanjutnya, luas masing-masing *bounding box* dihitung, dan luas gabungan (*union*) keduanya dihitung dengan mengurangi *area overlap* dari total luas kedua *box*. IoU dihitung sebagai rasio antara *area overlap* dan area gabungan, memberikan nilai antara 0 (tidak ada tumpang tindih) hingga 1 (sepenuhnya tumpang tindih). Fungsi ini berguna untuk menilai kualitas deteksi objek dalam aplikasi seperti pengenalan objek atau pelacakan.

8. Informasi jumlah kendaraan dan status lahan parkir

```
  # Menampilkan jumlah kendaraan dan status lahan parkir
         cv.putText(frame, f'Jumlah Mobil  : {jml}', (20, 50),
                 cv.FONT_HERSHEY_COMPLEX, 0.7, (0, 0, 255), 2)
         cv.putText(frame, f'Lahan Kosong (Hijau): {jumlah_hijau}', (20,
  80),
                 cv.FONT_HERSHEY_COMPLEX, 0.7, (0, 0, 255), 2)
```

```
            cv.putText(frame, f'Lahan Terisi (Merah): {jumlah_merah}', (20,
  110),
                    cv.FONT_HERSHEY_COMPLEX, 0.7, (0, 0, 255), 2)
            # Menampilkan jumlah kendaraan dan status lahan parkir
            cv.putText(frame, f'Jumlah lahan parkir  : {parkir}', (20, 140),
                    cv.FONT_HERSHEY_COMPLEX, 0.7, (0, 0, 255), 2)
            # Tulis frame yang sudah diproses ke video output
            cv.putText(frame, f'Koordinat : -8.5864943,116.0966586', (10, 50),
                    cv.FONT_HERSHEY_COMPLEX, 0.7, (0, 0, 255), 2)

            out.write(frame)
        # Tutup semua resources setelah proses selesai
        cap.release()
        out.release()
        return os.path.join(app.config['UPLOAD_FOLDER'], 'processed_video.mp4')
```

Kode di atas menambahkan informasi jumlah kendaraan dan status lahan parkir ke dalam *frame* video yang sedang diproses. Menggunakan fungsi cv.putText, teks seperti jumlah mobil yang terdeteksi (jml), jumlah lahan parkir kosong (jumlah_hijau), jumlah lahan terisi (jumlah_merah), total lahan parkir (parkir), dan koordinat lokasi parkir ditampilkan pada posisi tertentu dalam frame dengan font FONT_HERSHEY_COMPLEX, ukuran 0.7, dan warna merah. Setelah informasi ditambahkan, *frame* yang telah diproses ditulis ke *file* video keluaran menggunakan out.write(frame). Akhirnya, semua sumber daya video seperti *cap* dan *out* dilepaskan dengan release(), dan fungsi mengembalikan *path file* video yang telah diproses (processed_video.mp4). Kode ini memastikan bahwa hasil akhir berupa video dengan informasi tambahan terkait parkiran ditampilkan secara visual.

9. Mengonfigurasi dan mempersiapkan model deteksi objek

```
def getNet(self):
      # Panggil method colors
      COLORS=self.colorsRealtime()
      # Panggil method classNameRealtime
      class_name=self.classNameRealtime()
      # load model
      net=self.loadModel()
      # set net untuk CUDA target
      net.setPreferableBackend(cv.dnn.DNN_BACKEND_CUDA)
      net.setPreferableTarget(cv.dnn.DNN_TARGET_CUDA)
      # Deteksi model
      self.model= self.deteksiModel(net)
      return self.confthres,self.nmsthres,COLORS,class_name,self.model
```

Kode di atas mendefinisikan metode getNet dalam kelas Realtime yang digunakan untuk mengonfigurasi dan mempersiapkan model deteksi objek. Metode ini memanggil beberapa metode lain untuk mendapatkan informasi yang diperlukan, seperti colorsRealtime() untuk mendapatkan warna yang digunakan dalam visualisasi deteksi, dan classNameRealtime() untuk mendapatkan daftar kelas objek yang dapat dideteksi oleh model. Selanjutnya, model deteksi objek dimuat menggunakan loadModel(), dan pengaturan backend serta target untuk pemrosesan menggunakan CUDA dilakukan dengan net.setPreferableBackend(cv.dnn.DNN_BACKEND_CUDA) dan net.setPreferableTarget(cv.dnn.DNN_TARGET_CUDA), yang memungkinkan akselerasi GPU. Setelah itu, model deteksi objek diinisialisasi melalui metode deteksiModel(net). Akhirnya, metode getNet mengembalikan nilai ambang batas deteksi (confthres, nmsthres), warna, nama kelas, dan model deteksi yang telah disiapkan.

10. Membaca kelas oleh model YOLO

```
# Method untuk membaca nama class
    def classNameRealtime(self):
        self.class_name = []
```

```
        with open('Yolo/obj.names', 'r') as f:
            self.class_name = [cname.strip() for cname in f.readlines()]
        return self.class_name
```

Kode di atas mendefinisikan metode classNameRealtime yang digunakan untuk membaca nama kelas objek yang dapat dideteksi oleh model YOLO. Metode ini membuka *file* bernama 'Yolo/obj.names' yang berisi daftar nama kelas, dan kemudian membaca setiap baris dalam *file* tersebut. Setiap nama kelas diproses dengan menghilangkan karakter spasi atau *newline* di sekitarnya menggunakan strip(). Nama kelas yang telah diproses disimpan dalam atribut self.class_name, yang kemudian dikembalikan oleh metode ini. Dengan demikian, metode ini memberikan daftar nama kelas yang digunakan dalam deteksi objek oleh model YOLO.

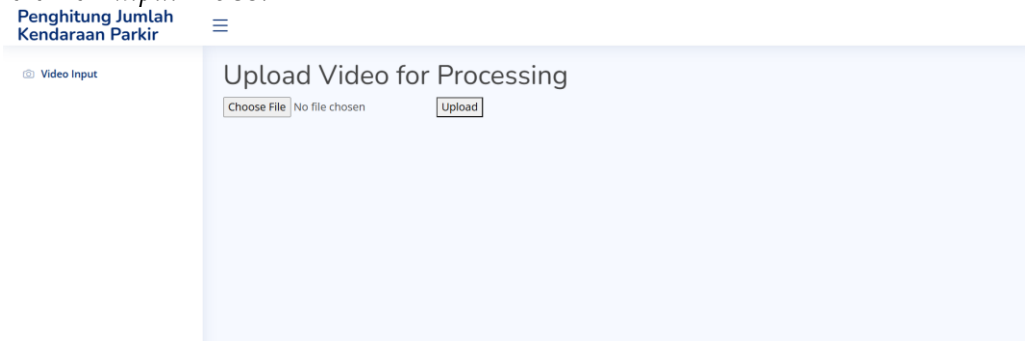
11. Menetapkan ukuran *input* dan parameter

```
# method untuk memberi  ukuran input ke model
    def deteksiModel(self,net):
        self.model = cv.dnn_DetectionModel(self.net)
      #   input ke model diatur 224 dan dilakukan scale normalisasi piksel
dengan 1/255 dan menukaran Saluran merah ke biru
        self.model.setInputParams(size=(224, 224), scale=1/255, swapRB=True)
        return self.model
```

Kode di atas mendefinisikan metode deteksiModel yang digunakan untuk mengonfigurasi model deteksi objek dengan menetapkan ukuran *input* dan parameter pengolahan gambar. Pertama, metode ini membuat objek cv.dnn_DetectionModel dengan model YOLOv4 yang telah dimuat sebelumnya (self.net). Selanjutnya, *input* ke model diatur dengan metode setInputParams, yang mengubah ukuran gambar menjadi 224x224 piksel, melakukan normalisasi piksel dengan skala 1/255 untuk menyesuaikan rentang nilai piksel, dan menukar urutan saluran warna dari format BGR (default OpenCV) menjadi RGB dengan parameter swapRB=True. Konfigurasi ini memastikan bahwa gambar yang dimasukkan ke model sesuai dengan persyaratan YOLOv4 dan siap untuk diproses. Metode ini kemudian mengembalikan objek model yang telah disiapkan untuk deteksi objek.

3.3. Pengujian Antarmuka Sistem

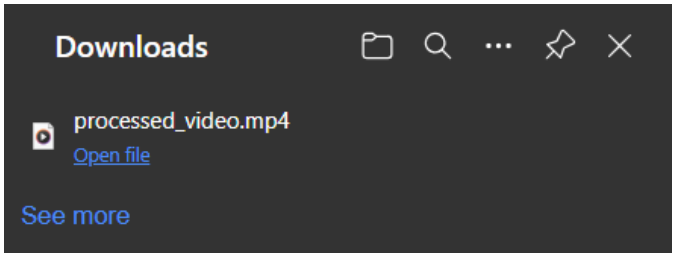
Setelah semua proses tersebut di implementasikan maka aplikasi berbasis *website* dapat digunakan. Adapun hasil perancangan aplikasi berbasis *website* ini dapat dilihat pada gambar 4.16 untuk halaman *input* video.



Gambar 6. Halaman *Input* Video

Pada gambar 6 merupakan halaman awal pada aplikasi untuk melakukan deteksi ketersediaan lahan parkir mobil. Pengguna terlebih dahulu memilih video yang ada di galeri dengan ekstensi mp4, jika video sudah dipilih pengguna kemudian menekan tombol *upload*. Setelah

video di *upload*, *system* akan memproses video yang telah di upload yang otomatis di *download* jika proses telah selesai dilakukan yang dapat dilihat pada gambar 7.



Gambar 7. *File* Hasil Deteksi

Selanjutnya, video hasil deteksi yang sudah di *download* dapat dibuka dan di jalankan diperangkat komputer seperti gambar 8.

Gambar 8. Isi File Hasil Deteksi

Pada gambar 8 merupakan hasil deteksi ketersediaan lahan parkir mobil yang dimana pada gambar tersebut terdapat jumlah mobil, lahan kosong, lahan terisi, jumlah lahan parkir, dan koordinat lokasi parkiran yang sesuai dengan google maps.

4. KESIMPULAN

1. Perangkat lunak berbasis *website* untuk mendeteksi dan menghitung mobil di parkiran dapat dibangun menggunakan YOLOv4 untuk deteksi objek. Aplikasi ini menggunakan *framework* seperti Flask untuk memproses video atau gambar dan menampilkan hasil deteksi, termasuk menghitung jumlah mobil yang terparkir dan menampilkan status lahan parkir.
2. Akurasi deteksi diukur dengan membandingkan hasil deteksi YOLOv4 dengan data *ground truth* menggunakan metrik *Intersection over Union* (IoU). Metrik ini membantu dalam mengevaluasi akurasi deteksi, termasuk mengidentifikasi kesalahan seperti *false positives* dan *false negatives*, untuk memastikan sistem memberikan hasil yang tepat.

DAFTAR PUSTAKA

[1] Arief, R. W. (2021). Sistem Deteksi dan Pengenalan Rambu Lalu Lintas Di Indonesia Mengunakan Algoritma YOLOv4= Traffic Sign Detection and Recognition System in Indonesia Using the YOLOv4 Algorithm (Doctoral dissertation, Universitas Hasanuddin).

[2] Budiarjo, D. D. (2020). IMPLEMENTASI SISTEM CERDAS PADA OTOMATISASI PENDETEKSIAN JENIS KENDARAAN DI JALAN RAYA.

[3] Ektrada, E., Hakim, L., & Kristanto, S. P. (2023). Sistem Tracking dan Counting Kendaraan Berbasis YOLO untuk Pemetaan Slot Parkir Kendaraan. Software Development, Digital Business Intelligence, and Computer Engineering, 1(02), 55-60.

[4] Fatmawati, F., & Narti, N. (2022). Perbandingan Algoritma C4. 5 dan Naive Bayes Dalam Klasifikasi Tingkat Kepuasan Mahasiswa Terhadap Pembelajaran Daring. JTIM: Jurnal Teknologi Informasi dan Multimedia, 4(1), 1-12.

[5] Hudaya, M. A., Santoso, I., & Soetrisno, Y. A. A. (2020). Perancangan Sistem Pelacakan (Tracking) Dan Perhitungan Kendaraan Pada Citra Bergerak Menggunakan Metode Convolutional Neural Network. Transient: Jurnal Ilmiah Teknik Elektro, 9(1), 80-87.

[6] Jalled, F., & Voronkov, I. (2016). Object Detection using Image Processing. 1–6. Retrieved from http://arxiv.org/abs/1611.07791.

[7] M. Singh, A. Verma, A. Parasher, N. Chauhan, and G. Budhiraja, "Implementation of Database Using Python Flask Framework," vol. 8, no. 12, pp. 24894–24899, 2019, doi: 10.18535/ijecs/v8i12.4399.

[8] Marutho, D. (2019). Perbandingan Metode Naive Bayes, KNN, Decision Tree Pada Laporan Water Level Jakarta. Jurnal Ilmiah Infokam, 15(2).

[9] Menegaz, M. 2018. Understanding YOLO – Hacker Noon. Diambil kembali dari HackerNoon: https://hackernoon.com/understanding-yolo-f5a74bbc7967.

[10] Mulyanto, A., Jatmiko, W., Mursanto, P., Prasetyawan, P., & Borman, R. I. (2021). A New Indonesian Traffic Obstacle Dataset and Performance Evaluation of YOLOv4 for ADAS. Journal of ICT Research & Applications, 14(3).

[11] Nurhawaisah, N. (2021). Rancang Bangun Sistem Klasifikasi dan Counting Kendaraan dengan Metode YOLOv3 (Doctoral dissertation, Institut Teknologi Kalimantan).

[12] Nurhikmat, T. (2018). Implementasi deep learning untuk image classification menggunakan algoritma Convolutional Neural Network (CNN) pada citra wayang golek.

[13] O'Shea, K., & Nash, R. 2015. An Introduction to Convolutional Neural Networks. Diambil kembali dari arXiv: https://arxiv.org/abs/1511.08458

[14] Oklilas, A. F., Dwinta, D., Shofi, G., Mariza, N. P., Kinanti, S. A., & Sari, Y. A. (2023). Akurasi Pengujian Model Hasil Training menggunakan YOLOv4 untuk Pengenalan Kendaraan di Jalan Raya. JUPITER: Jurnal Penelitian Ilmu dan Teknologi Komputer, 15(1d), 799-806.

[15] Pandia, M. (2024). Kajian Literatur Multimedia Retrieval: Machine Learning Untuk Pengenalan Wajah. Jurnal Ilmu Komputer dan Sistem Informasi (JIKOMSI), 7(1), 161-166.

[16] Prabowo, W. A., & Wiguna, C. (2021). Sistem informasi UMKM bengkel berbasis web menggunakan metode scrum. Jurnal Media Informatika Budidarma, 5(1), 149-156.

[17] Putra, A. K., & Bunyamin, H. (2020). Pengenalan Simbol Matematika dengan Metode Convolutional Neural Network (CNN). Jurnal STRATEGI-Jurnal Maranatha, 2(2), 426-433.

[18] Rahmawati, L., & Adi, K. (2017). Rancang bangun penghitung dan pengidentifikasi kendaraan menggunakan Multiple Object Tracking. Youngster Physics Journal, 6(1), 70-75.

[19] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. arXiv, 1-10.

[20] Suwito, S. (2021). Penggunaan Bahasa Pemrograman Python untuk Membuat Aplikasi yang Dapat Digunakan Sebagai Media Dalam Mengajarkan Konsep Perilaku Lentur Balok Beton Bertulang.

[21] V. R. Vyshnavi and A. Malik, "Efficient Way of Web Development Using Python and Flask," vol. 6, no. 2, pp. 16–19, 2019.

[22] Valentina, R., Rostianingsih, S., & Tjondrowiguno, A. N. (2020). Pengenalan Gambar Botol Plastik dan Kaleng Minuman Menggunakan Metode Convolutional Neural Network. Jurnal Infra, 8(1), 249-254.