

IF4073 Pemrosesan Citra Digital

DOKUMEN TUGAS 4:

PENGENALAN JENIS KENDARAAN DALAM CITRA MENGGUNAKAN METODE PEMBELAJARAN MESIN TRADISIONAL DAN MODERN



Oleh:

13521042 Kevin John Wesley Hutabarat

13521107 Jericho Russel Sebastian

13521140 Ryan Samuel Chandra

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024**

DAFTAR ISI

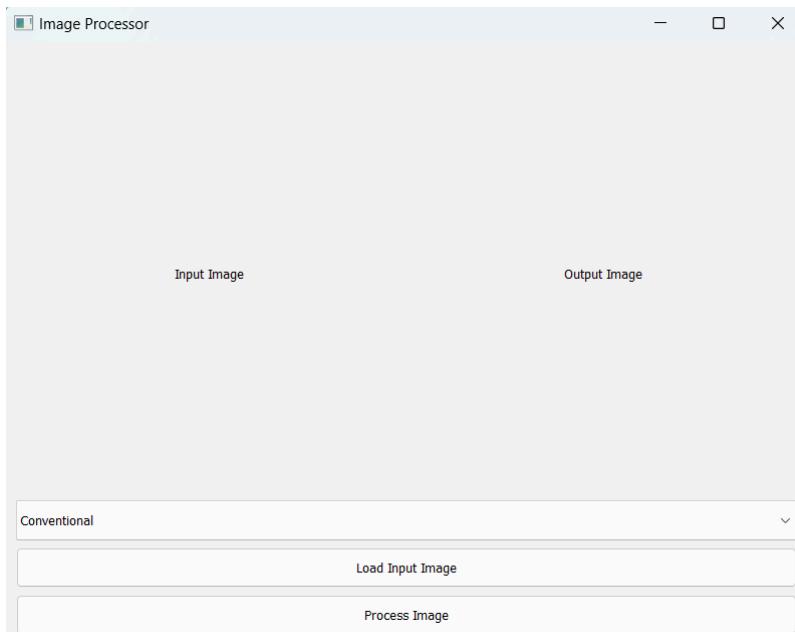
DAFTAR ISI.....	2
DAFTAR GAMBAR.....	3
I. Hasil Tangkap Layar GUI.....	4
II. Deteksi Kendaraan dengan Pendekatan Konvensional.....	4
III. Deteksi Kendaraan dengan Pendekatan Deep Learning Menggunakan CNN.....	10
IV. Kesimpulan.....	15
V. Komentar dan Refleksi.....	15
VI. Alamat GitHub.....	15

DAFTAR GAMBAR

Gambar 1.1 Tampilan aplikasi.....	4
Gambar 2.1 Hasil pengujian untuk citra 1 dengan pendekatan konvensional.....	5
Gambar 2.2 Hasil pengujian untuk citra 2 dengan pendekatan konvensional.....	6
Gambar 2.3 Hasil pengujian untuk citra 3 dengan pendekatan konvensional.....	6
Gambar 2.4 Hasil pengujian untuk citra 4 dengan pendekatan konvensional.....	7
Gambar 2.5 Hasil pengujian untuk citra 5 dengan pendekatan konvensional.....	7
Gambar 2.6 Hasil pengujian untuk citra 6 dengan pendekatan konvensional.....	8
Gambar 2.7 Hasil pengujian untuk citra 7 dengan pendekatan konvensional.....	8
Gambar 2.8 Hasil pengujian untuk citra 8 dengan pendekatan konvensional.....	9
Gambar 2.9 Hasil pengujian untuk citra 9 dengan pendekatan konvensional.....	9
Gambar 2.10 Hasil pengujian untuk citra 10 dengan pendekatan konvensional.....	10
Gambar 2.12 Hasil pengujian untuk citra 2 dengan pendekatan deep learning.....	11
Gambar 2.13 Hasil pengujian untuk citra 3 dengan pendekatan deep learning.....	12
Gambar 2.14 Hasil pengujian untuk citra 4 dengan pendekatan deep learning.....	12
Gambar 2.15 Hasil pengujian untuk citra 5 dengan pendekatan deep learning.....	13
Gambar 2.16 Hasil pengujian untuk citra 6 dengan pendekatan deep learning.....	13
Gambar 2.17 Hasil pengujian untuk citra 7 dengan pendekatan deep learning.....	14
Gambar 2.18 Hasil pengujian untuk citra 8 dengan pendekatan deep learning.....	14
Gambar 2.19 Hasil pengujian untuk citra 9 dengan pendekatan deep learning.....	15
Gambar 2.20 Hasil pengujian untuk citra 10 dengan pendekatan deep learning.....	15

I. Hasil Tangkap Layar GUI

Berikut adalah hasil tangkapan layar dari GUI aplikasi ini



Gambar 1.1 Tampilan aplikasi

GUI diimplementasikan pada python dengan pustaka PyQt5. Pustaka ini memungkinkan pengembang untuk membuat antarmuka dengan menyediakan fungsi-fungsi siap pakai untuk membuat setiap komponennya. Terdapat komponen dropdown untuk memilih fungsi pendekripsi objek yang ingin digunakan (konvensional atau deep learning). Pengguna dapat memasukkan gambar dengan menekan tombol “Load Input Image”, kemudian memprosesnya dengan tombol “Process Image”. Hasil gambar dari pemrosesan akan muncul pada komponen “Output Image”.

II. Deteksi Kendaraan dengan Pendekatan Konvensional

1. Kode program

Berikut adalah implementasi kode program segmentasi, ekstraksi fitur, prediksi, dan visualisasi untuk deteksi jenis kendaraan dengan pendekatan konvensional/tradisional.

```
import cv2
from skimage.feature import hog

def extract_features(segmented_image):
    gray = cv2.cvtColor(segmented_image, cv2.COLOR_BGR2GRAY)
    resized = cv2.resize(gray, (128, 128)) # Resize gambar ke ukuran tetap

    # Ekstraksi fitur HOG
    hog_features = hog(resized, orientations=9, pixels_per_cell=(8, 8),
```

```

        cells_per_block=(2, 2), block_norm='L2-Hys',
        visualize=False)
    return (hog_features)

import cv2
import numpy as np

def segment_image(image_path):
    #Reference:
    https://www.analyticsvidhya.com/blog/2021/09/image-segmentation-algorithms-with-implementation-in-python/

    img = cv2.imread(image_path)
    b, g, r = cv2.split(img)
    rgb_img = cv2.merge([r, g, b]) # Convert BGR to RGB for display

    # Convert to grayscale and apply Gaussian blur
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Apply Otsu's thresholding
    ret, thresh = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY_INV
+ cv2.THRESH_OTSU)

    # Noise removal
    kernel = np.ones((3, 3), np.uint8)
    closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel,
iterations=2)

    # Sure background area
    sure_bg = cv2.dilate(closing, kernel, iterations=3)

    # Finding sure foreground area
    dist_transform = cv2.distanceTransform(sure_bg, cv2.DIST_L2, 5)
    ret, sure_fg = cv2.threshold(dist_transform, 0.5 *
dist_transform.max(), 255, 0)

    # Finding unknown region
    sure_fg = np.uint8(sure_fg)
    unknown = cv2.subtract(sure_bg, sure_fg)

    # Marker labeling
    ret, markers = cv2.connectedComponents(sure_fg)

    # Add one to all labels so that sure background is not 0, but 1
    markers = markers + 1
    markers[unknown == 255] = 0

    # Watershed algorithm
    markers = cv2.watershed(img, markers)
    img[markers == -1] = [255, 0, 0] # Mark the boundaries

    # Extract the segmented image using the threshold mask
    segmented_image = img.copy() # Start with a copy of the original

```

```

image
    segmented_image[thresh == 0] = [0, 0, 0] # Set the background to
black

    return (thresh, segmented_image)

import cv2
import numpy as np
import matplotlib.pyplot as plt

def visualize_bboxes(thresh_image, original_image, classname):
    y_non_zero, x_non_zero = np.nonzero(thresh_image)

    if len(x_non_zero) > 0 and len(y_non_zero) > 0:
        x_min, x_max = x_non_zero.min(), x_non_zero.max()
        y_min, y_max = y_non_zero.min(), y_non_zero.max()

        # Gambar bounding box pada gambar asli
        cv2.rectangle(original_image, (x_min, y_min), (x_max, y_max), (0,
0, 255), 2)

        # Tentukan posisi teks
        text_x = x_min
        text_y = y_min - 10

        if text_y < 10:
            text_y = y_max - 10
            text_x = text_x + 20

        # Tambahkan label
        cv2.putText(original_image, classname, (text_x, text_y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

    return (original_image)

import cv2
from segment_image import segment_image
from extract_features import extract_features
from visualize_bboxes import visualize_bboxes
import matplotlib.pyplot as plt

import joblib
model = joblib.load("svm_vehicle_detection_model.pkl")

def predict(image_path):
    class_names = ['Ambulance', 'Bus', 'Car', 'Motorcycle', 'Truck']

    thresh_image, segmented_image = segment_image(image_path)
    features = extract_features(segmented_image)
    prediction = model.predict([features])[0]
    classname = class_names[prediction]

    original_image = cv2.imread(image_path)
    detected = visualize_bboxes(thresh_image, original_image,

```

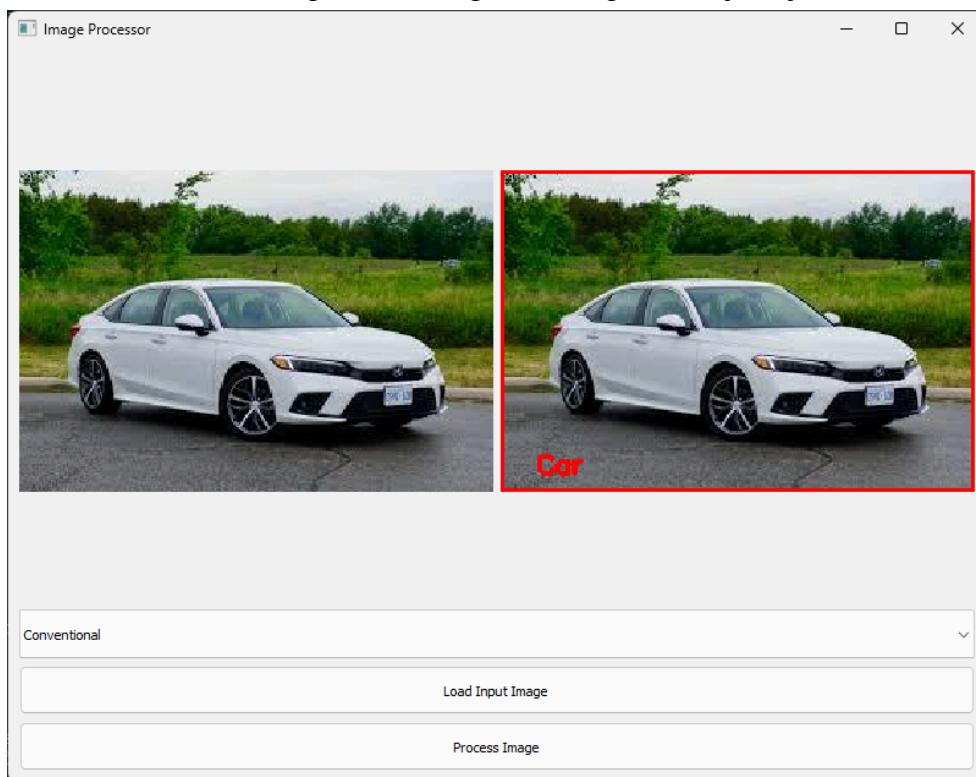
```
classname)
    return(detected)

image = predict("test.jpg")
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

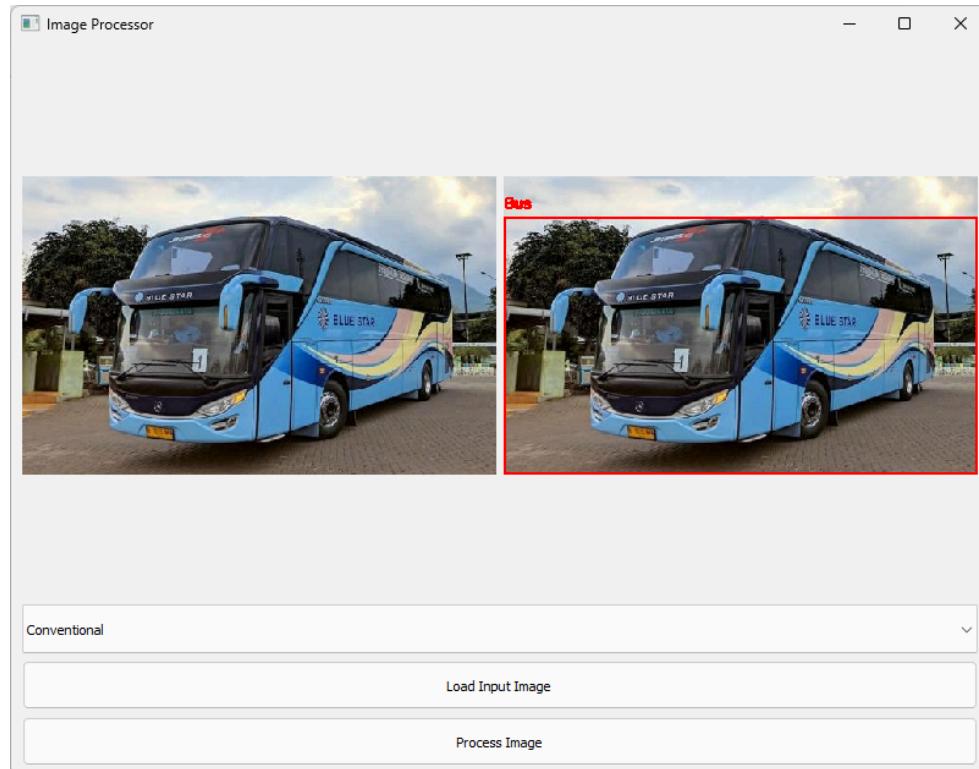
Algoritma yang digunakan adalah SVM (*Support Vector Machine*). Model ini dilatih dengan fitur terekstraksi dari 80% keseluruhan *dataset*, untuk dapat memprediksi dengan tepat. Pembangunan model dapat dilihat pada file “ipynb” di repositori Github.

2. Contoh Eksekusi

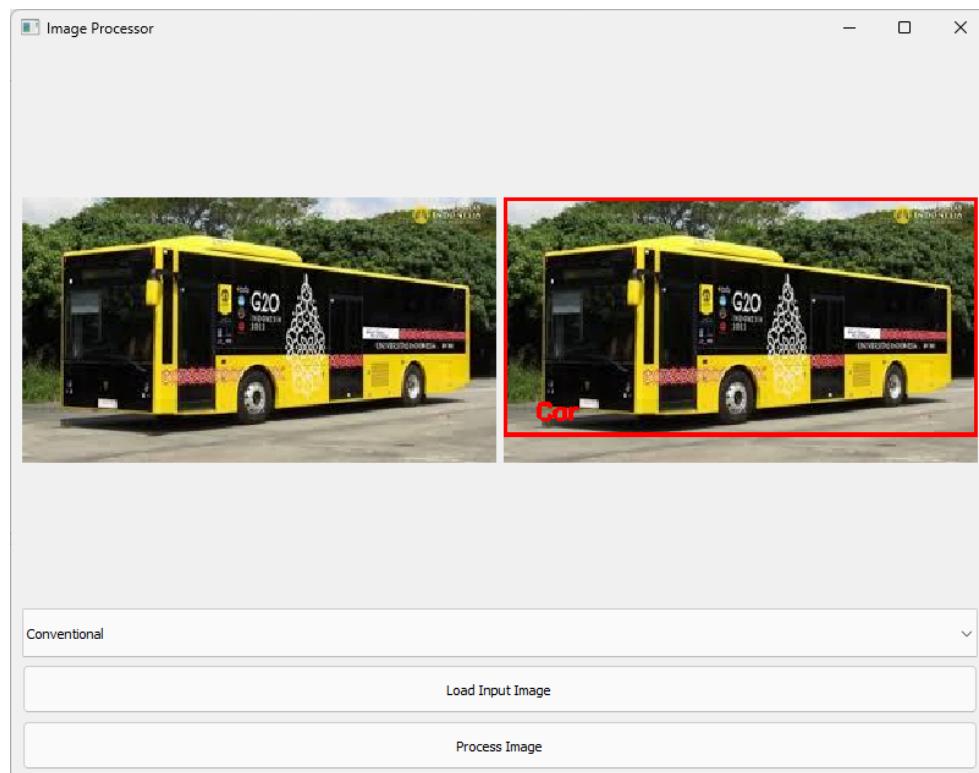
Berikut adalah hasil eksperimen dengan beberapa citra uji wajib dan tambahan



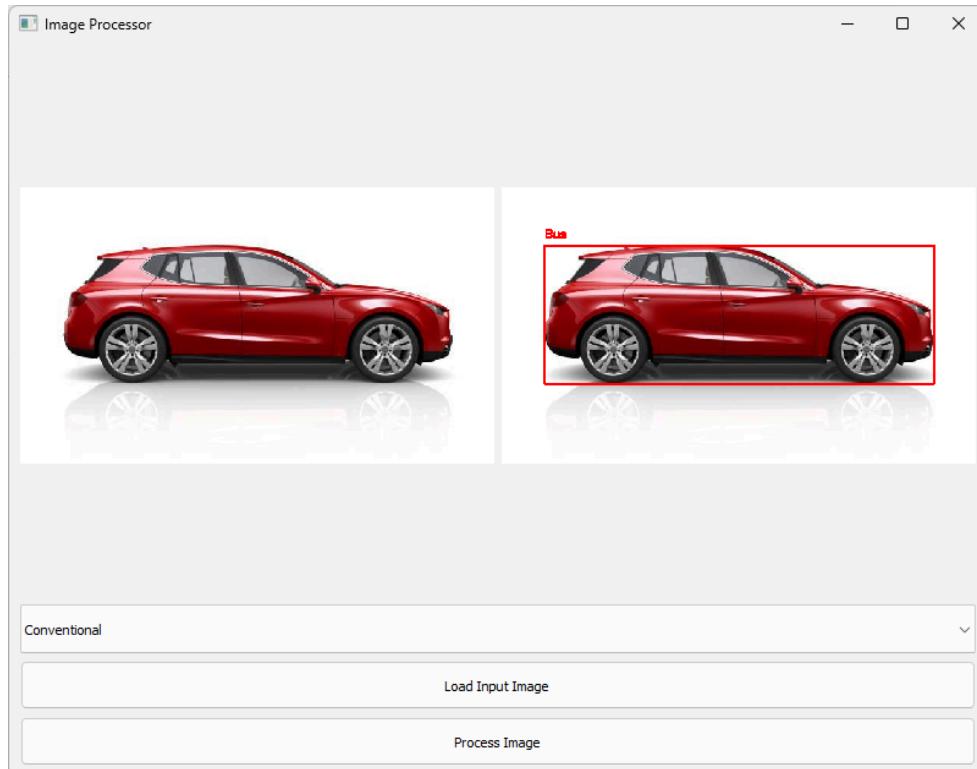
Gambar 2.1 Hasil pengujian untuk citra 1 dengan pendekatan konvensional



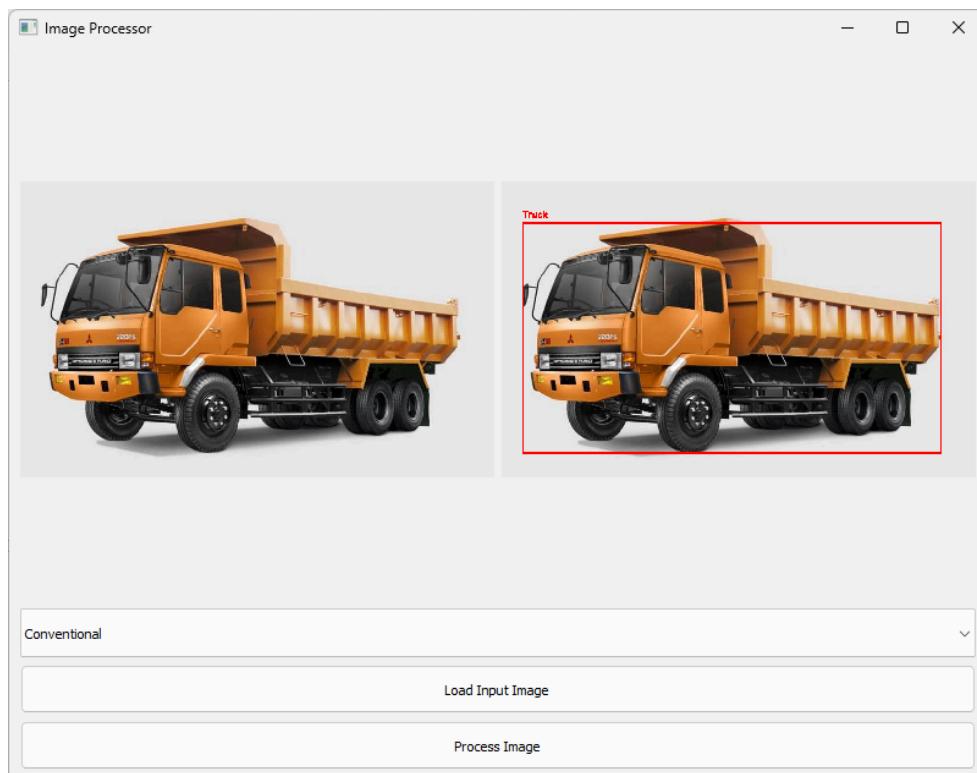
Gambar 2.2 Hasil pengujian untuk citra 2 dengan pendekatan konvensional



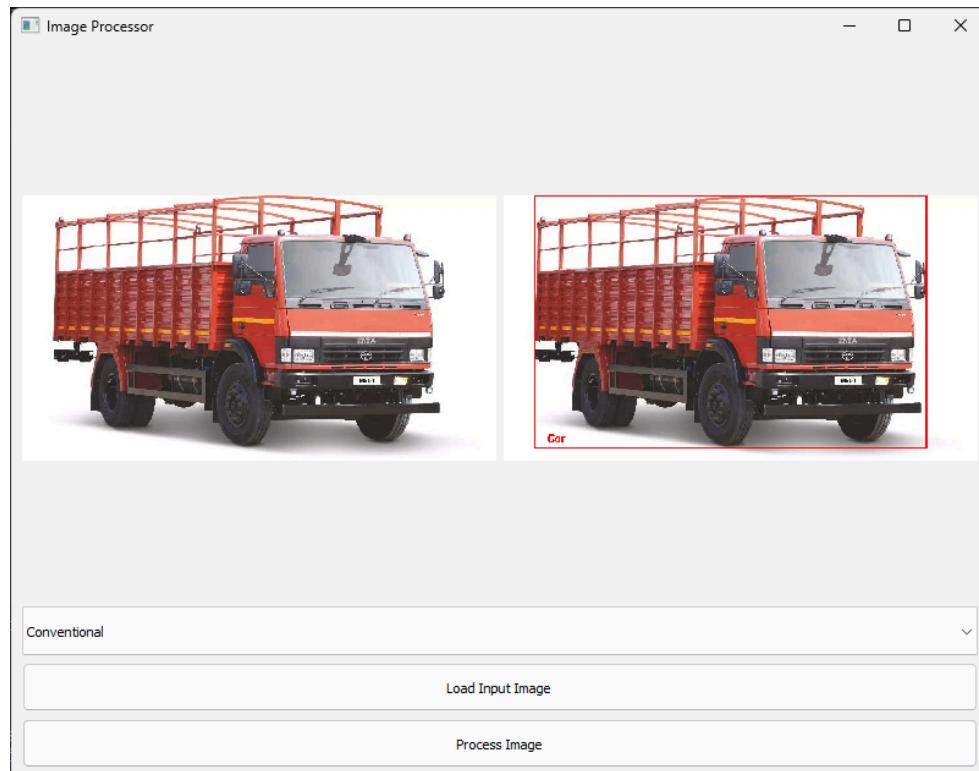
Gambar 2.3 Hasil pengujian untuk citra 3 dengan pendekatan konvensional



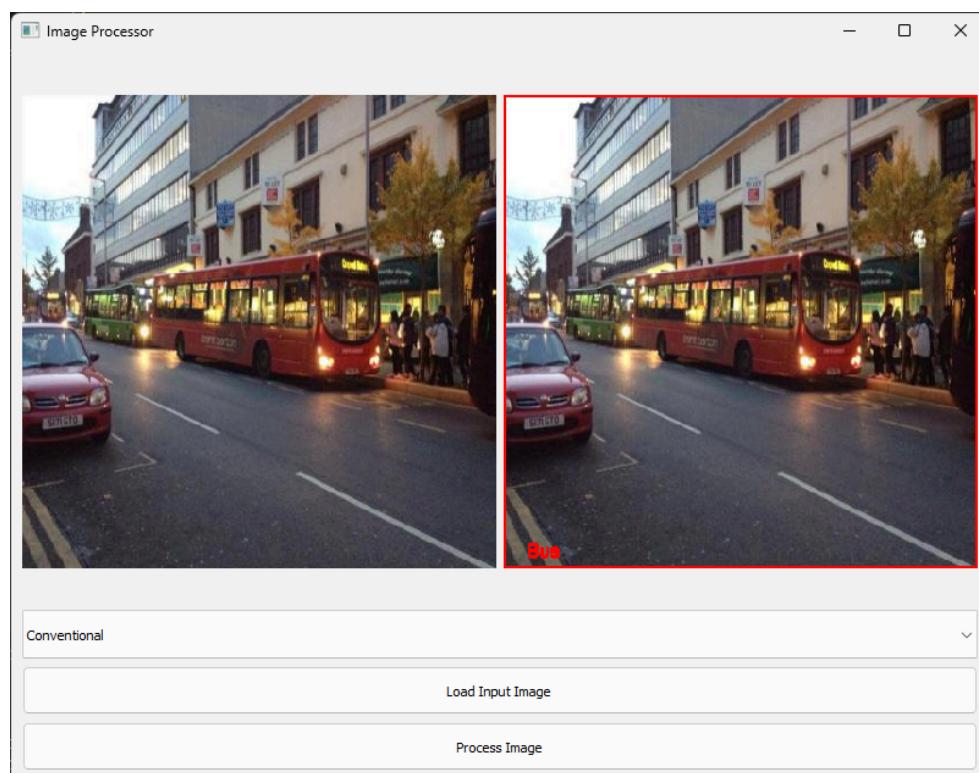
Gambar 2.4 Hasil pengujian untuk citra 4 dengan pendekatan konvensional



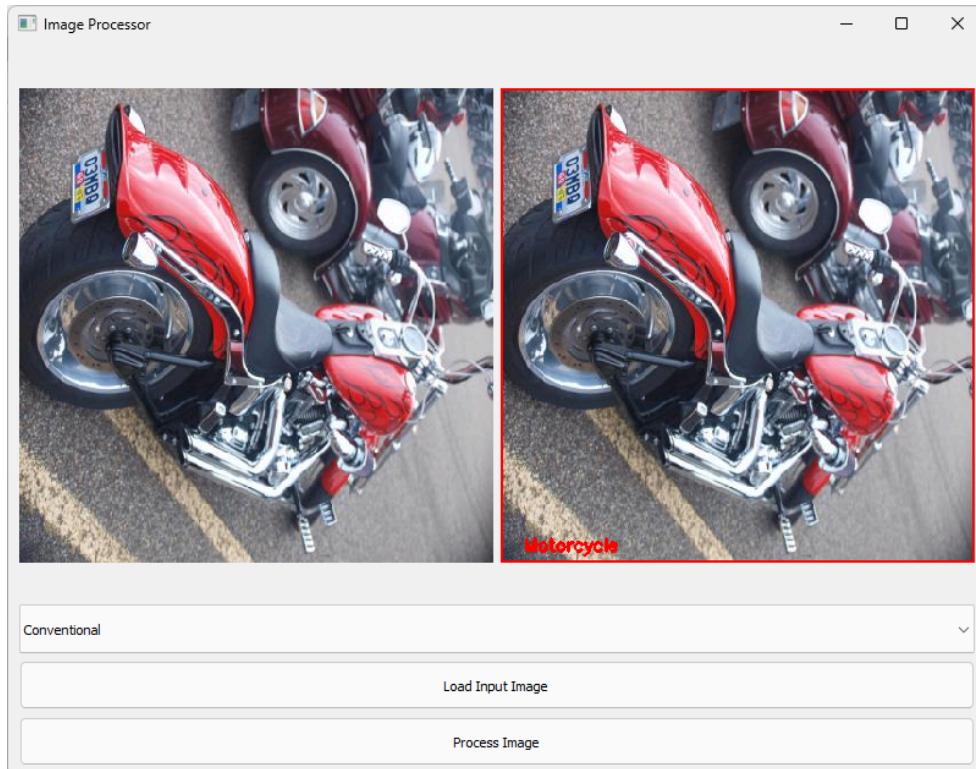
Gambar 2.5 Hasil pengujian untuk citra 5 dengan pendekatan konvensional



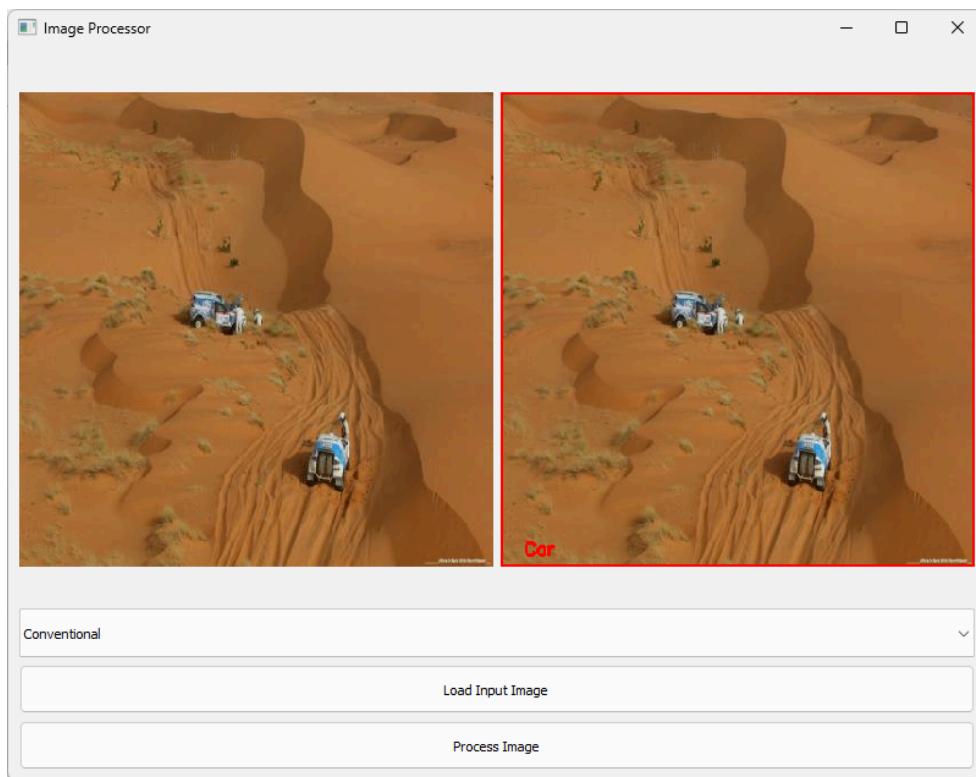
Gambar 2.6 Hasil pengujian untuk citra 6 dengan pendekatan konvensional



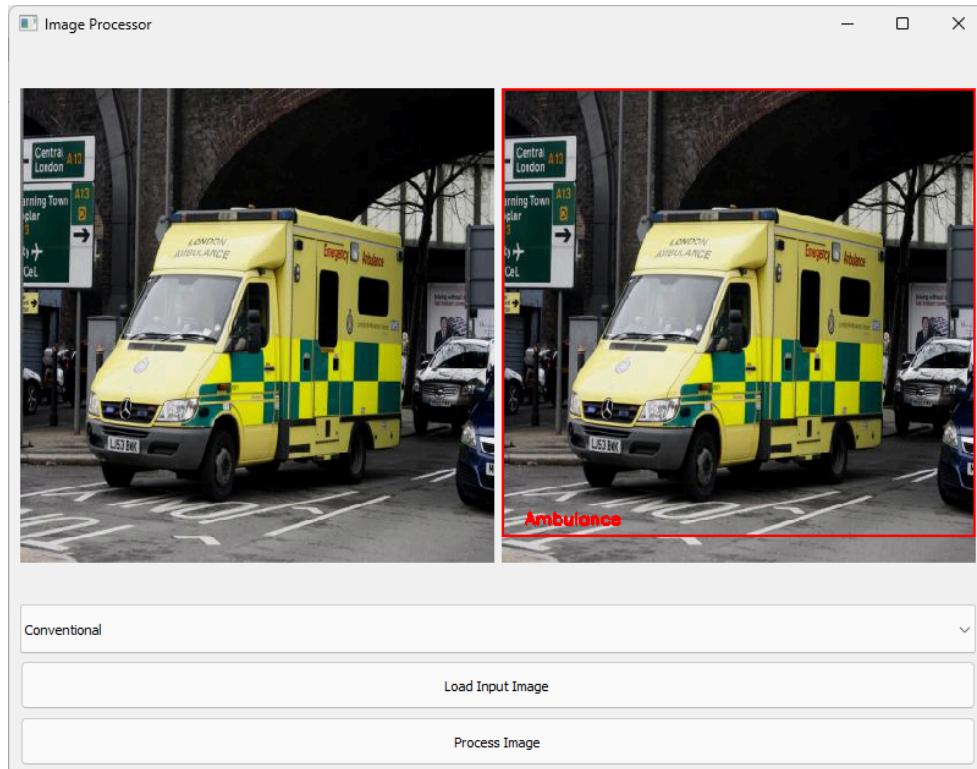
Gambar 2.7 Hasil pengujian untuk citra 7 dengan pendekatan konvensional



Gambar 2.8 Hasil pengujian untuk citra 8 dengan pendekatan konvensional



Gambar 2.9 Hasil pengujian untuk citra 9 dengan pendekatan konvensional



Gambar 2.10 Hasil pengujian untuk citra 10 dengan pendekatan konvensional

3. Analisis

Deteksi kendaraan dengan algoritma pembelajaran mesin tradisional, SVM, sangat bergantung pada hasil segmentasi citra. Sayangnya, apabila gambar terlalu “berisik” karena ada pohon, manusia, atau objek lainnya yang juga dominan, segmentasi menggunakan metode *Otsu’s threshold* memberikan hasil yang kurang baik. Dapat dilihat bahwa *bounding box* yang dihasilkan juga tidak tepat pada objek. Hal ini dikarenakan *bounding box* digambar pada titik-titik maksimum dan minimum objek hasil segmentasi, yang memang kurang tepat.

Objek yang berhasil didapatkan dari segmentasi yang akan diekstrak menjadi fitur. Algoritma ekstraksi fitur juga sangat berpengaruh. Dalam hal ini, algoritma yang digunakan adalah HOG (*Histogram of Oriented Gradients*) dari pustaka *scikit-image*.

Model SVM yang dibangun dapat memprediksi dengan benar pada sebagian besar kasus, dengan akurasi 84,86% terhadap data uji. Kesalahan mungkin terjadi karena kemiripan antara bentuk mobil, bus, dan truk.

III. Deteksi Kendaraan dengan Pendekatan *Deep Learning* Menggunakan CNN

1. Kode program

Berikut adalah implementasi kode program untuk deteksi kendaraan dengan pendekatan *deep learning* menggunakan model OWL-ViT.

```
from transformers import OwlViTProcessor, OwlViTForObjectDetection
import torch
import cv2

model_path = 'google/owlvit-base-patch16'
label_texts = ['Ambulance', 'Bus', 'Car', 'Motorcycle', 'Truck']

class DeepLearningProcessor:
    def __init__(self):
        # Load feature extractor and pre-trained model
        self._proc = OwlViTProcessor.from_pretrained(model_path)
        self._model = OwlViTForObjectDetection.from_pretrained(
            model_path,
            ignore_mismatched_sizes=True,
        )

    def process(self, img_in: cv2.typing.MatLike):
        # Send input image to model
        inp = self._proc(text=label_texts, images=img_in,
return_tensors='pt')
        out = self._model(**inp)

        # Thresholding and bounding box denormalization
        h, w, _ = img_in.shape
        target_sizes = [(h, w)]
        res = self._proc.post_process_object_detection(outputs=out,
threshold=0.1, target_sizes=target_sizes)

        # Draw bounding box for detected objects on output image
        img_out = img_in.copy()
        labels, boxes = res[0]['labels'], res[0]['boxes']
        for label_idx, box in zip(labels, boxes):
            label_text = label_texts[label_idx]
            box: torch.Tensor = box.round().int()
            [x1, y1, x2, y2] = box.tolist()
            print(f'Detected {label_text} at ({x1}, {y1}) : ({x2}, {y2})')

            cv2.rectangle(img_out, (x1, y1), (x2, y2), (0, 0, 255), 2)

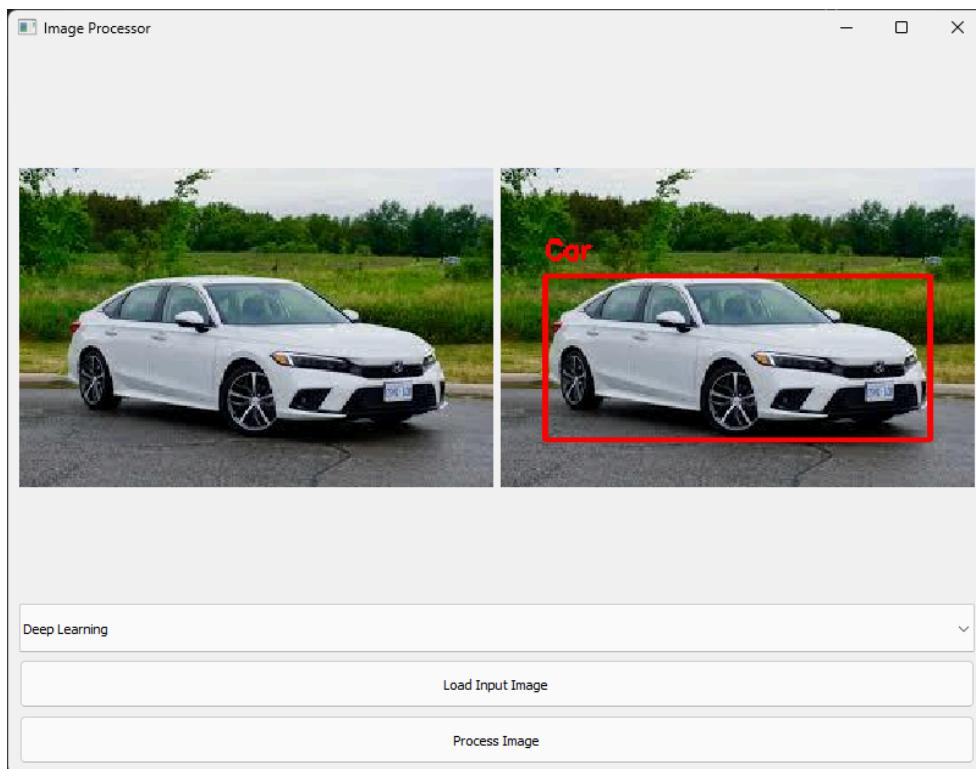
            text_x = x1
            text_y = y1 - 10

            if text_y < 10:
                text_y = y2 - 10
                text_x = text_x + 20
```

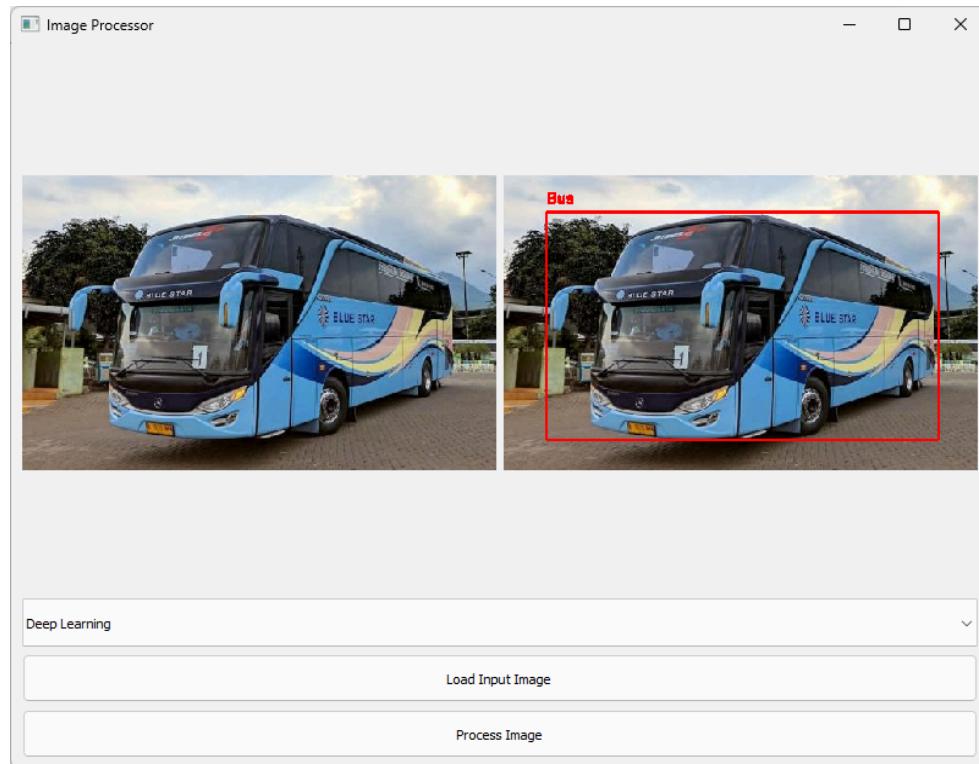
```
        cv2.putText(img_out, label_text, (text_x, text_y),  
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)  
  
    return img_out
```

2. Contoh Eksekusi

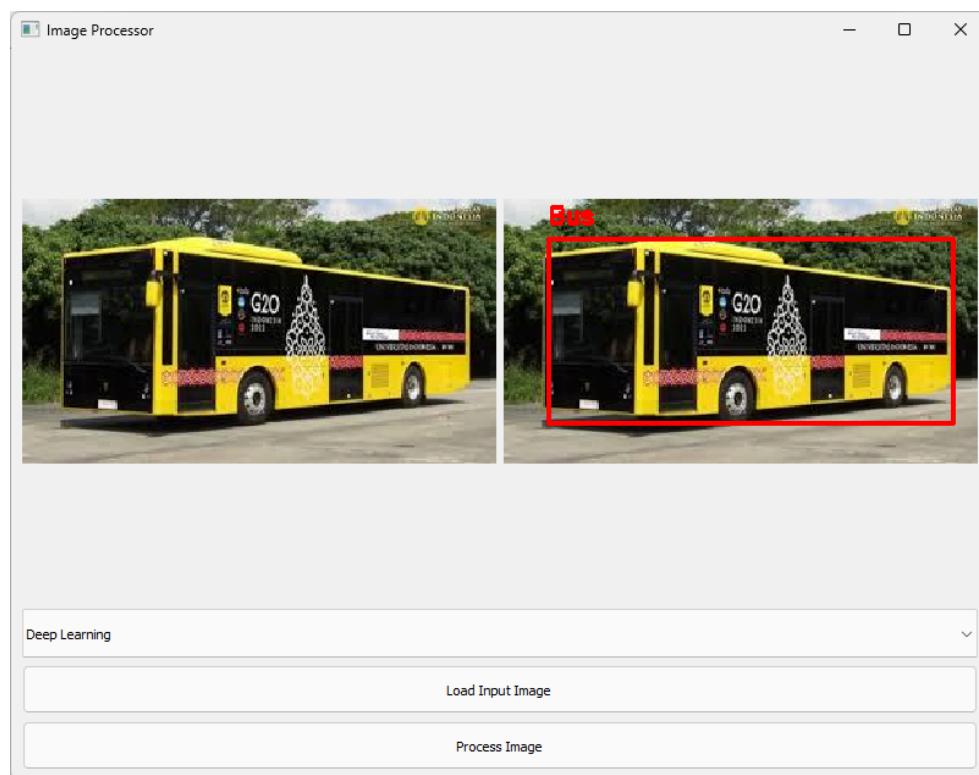
Berikut adalah hasil eksperimen dengan beberapa citra uji wajib dan tambahan



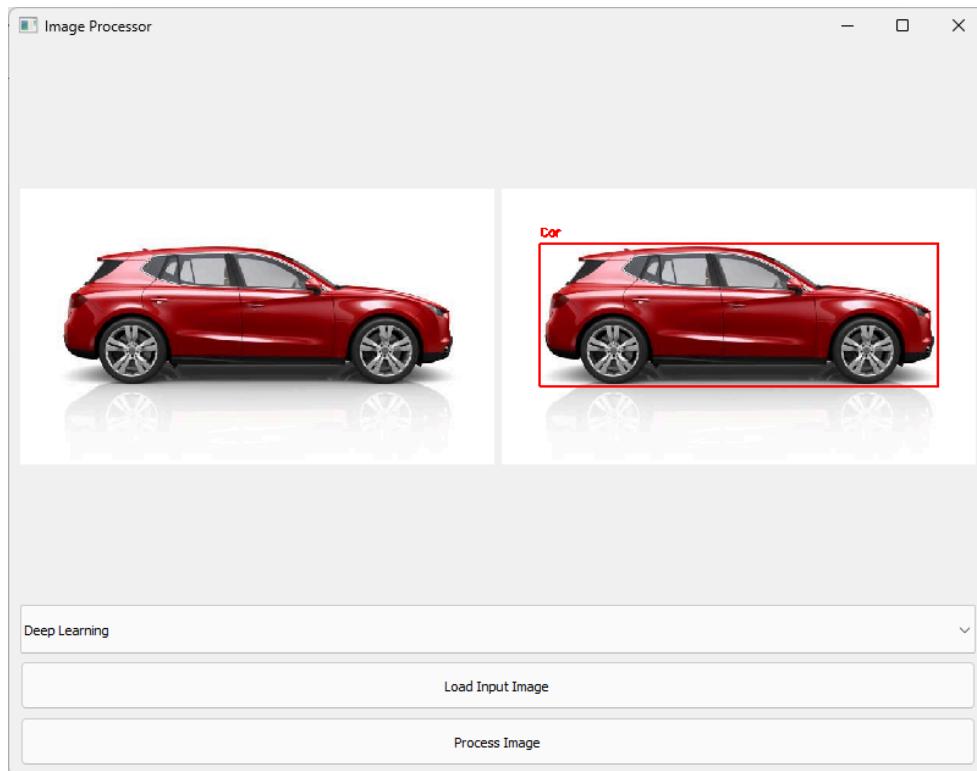
Gambar 2.11 Hasil pengujian untuk citra 1 dengan pendekatan *deep learning*



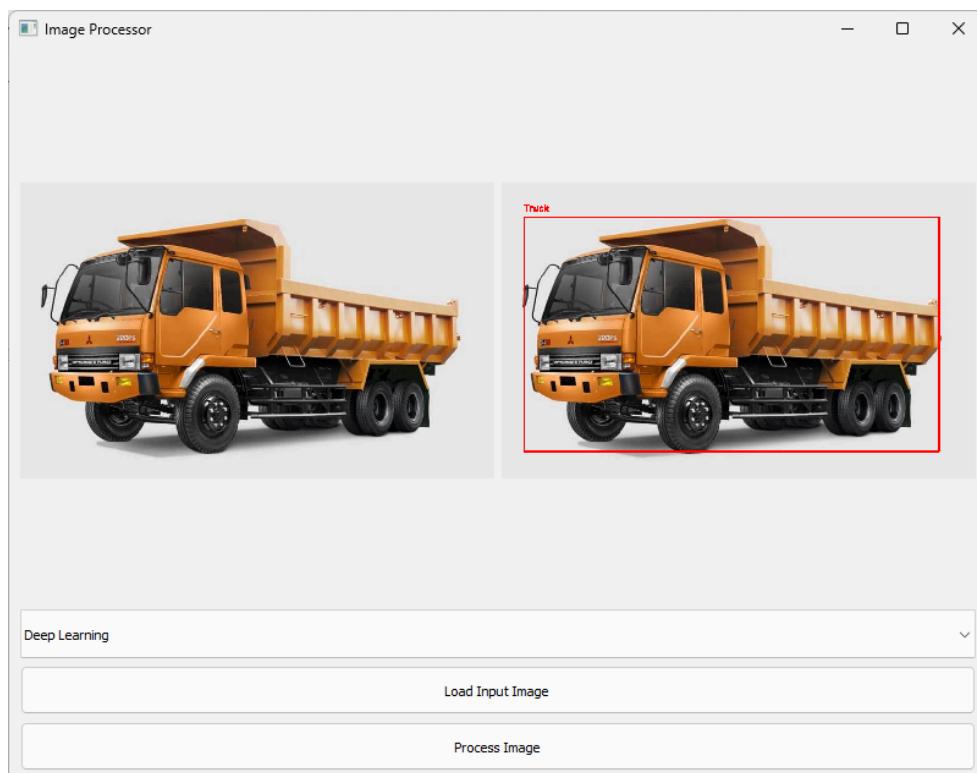
Gambar 2.12 Hasil pengujian untuk citra 2 dengan pendekatan *deep learning*



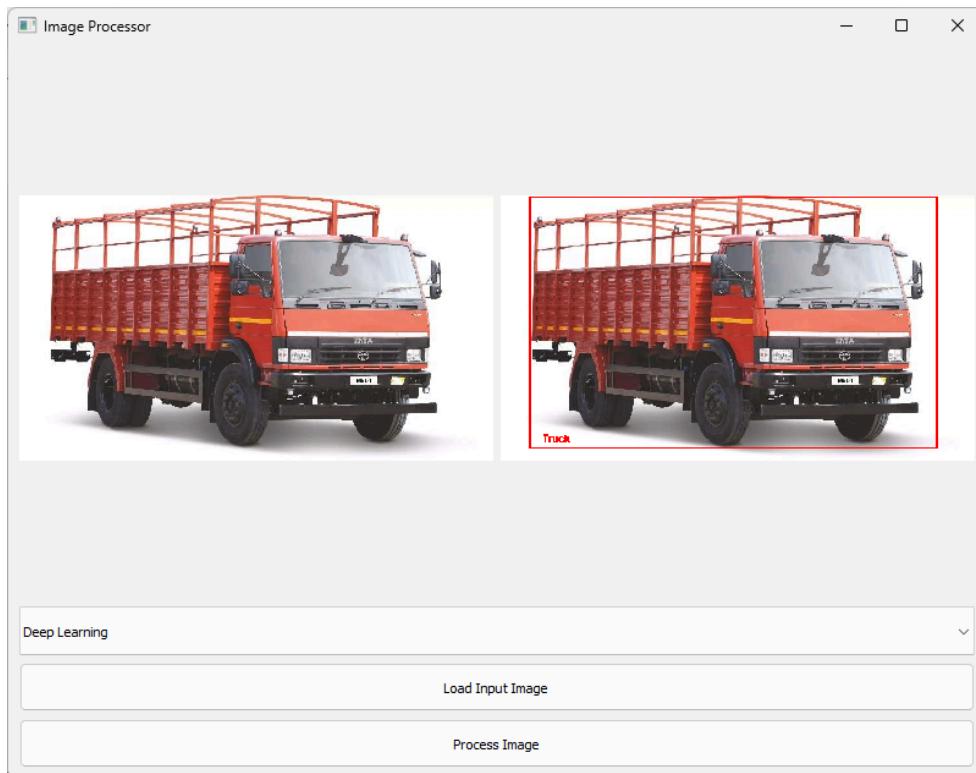
Gambar 2.13 Hasil pengujian untuk citra 3 dengan pendekatan *deep learning*



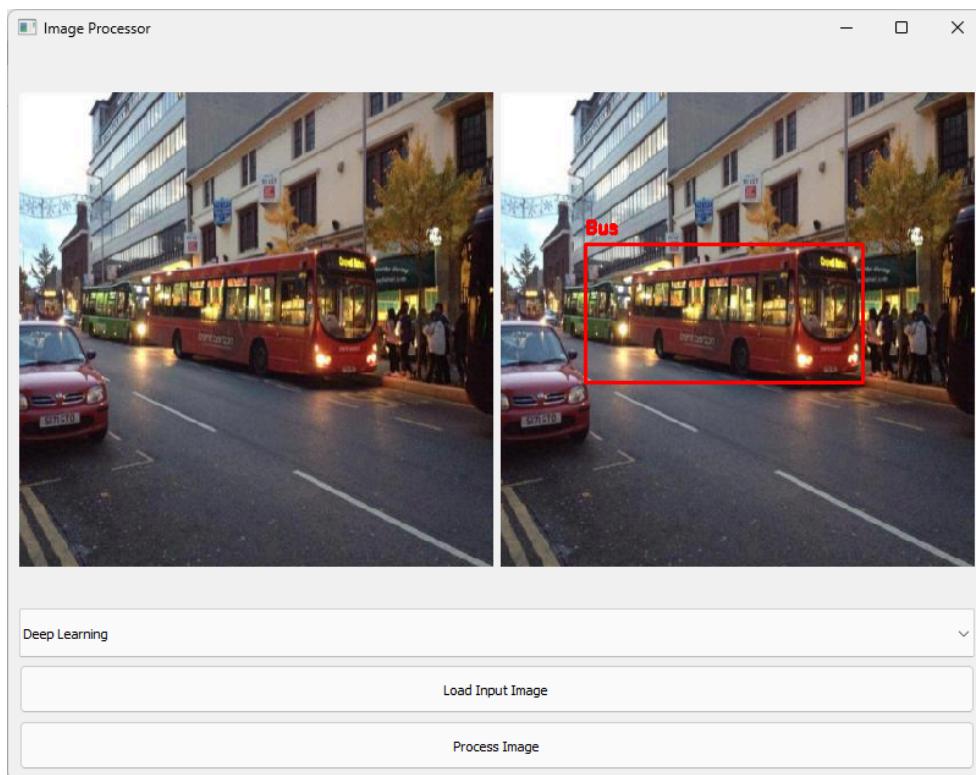
Gambar 2.14 Hasil pengujian untuk citra 4 dengan pendekatan *deep learning*



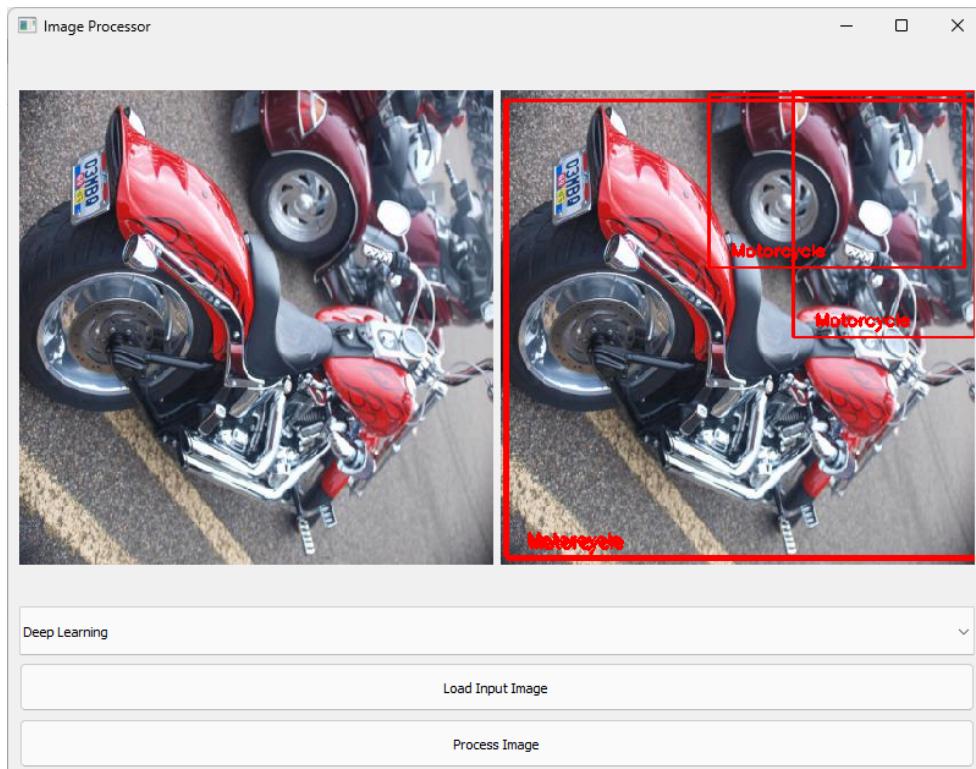
Gambar 2.15 Hasil pengujian untuk citra 5 dengan pendekatan *deep learning*



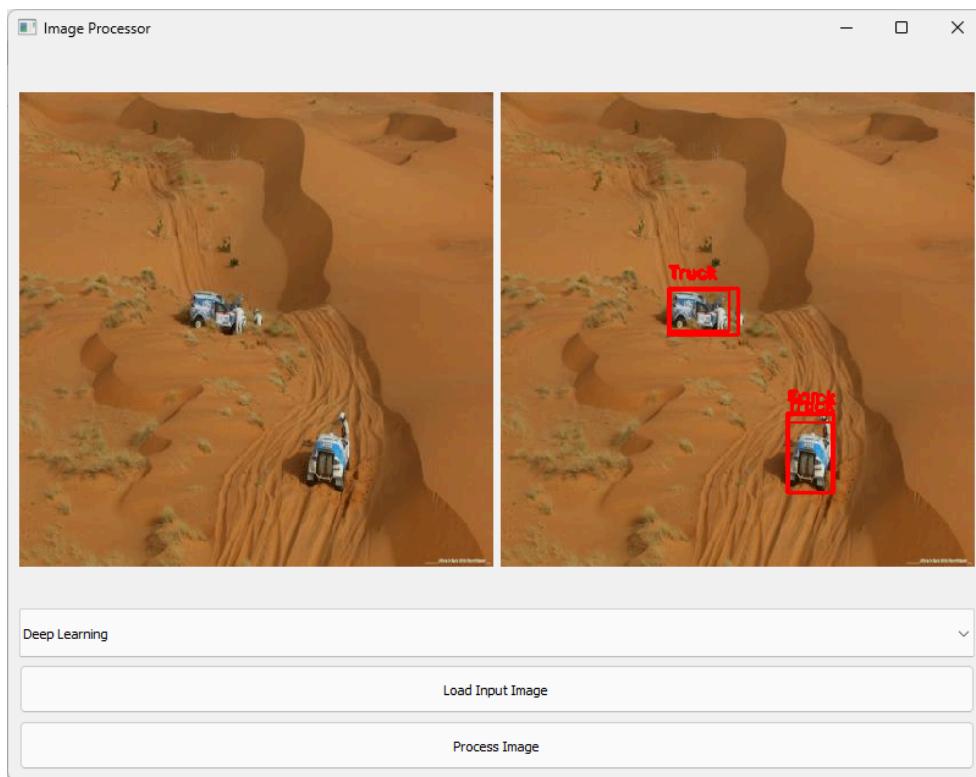
Gambar 2.16 Hasil pengujian untuk citra 6 dengan pendekatan *deep learning*



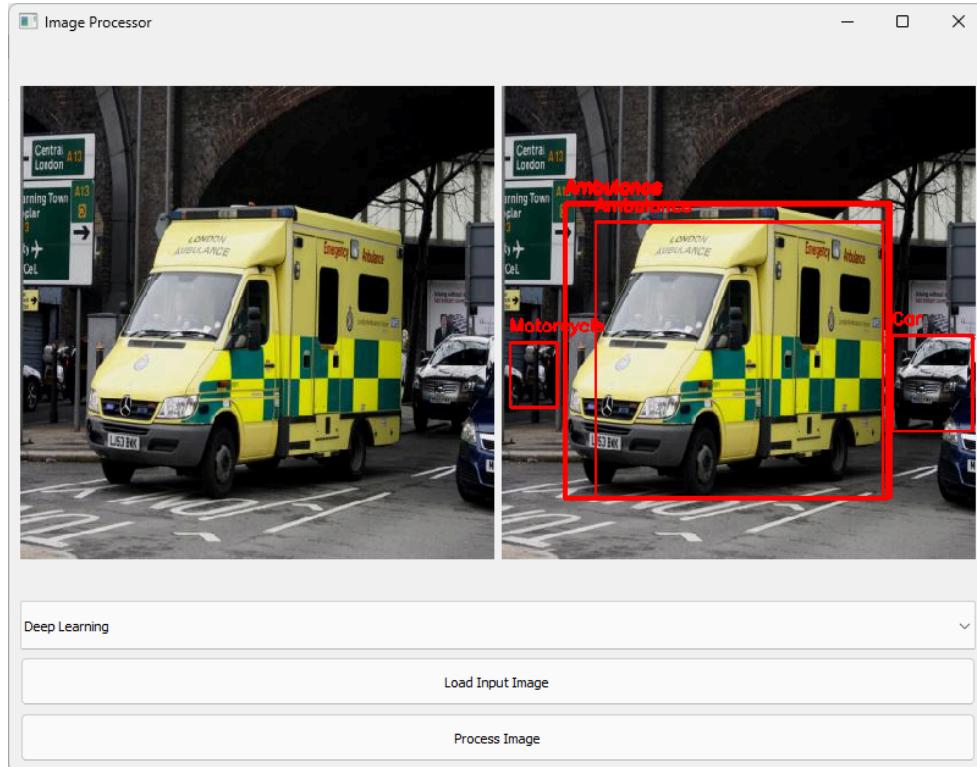
Gambar 2.17 Hasil pengujian untuk citra 7 dengan pendekatan *deep learning*



Gambar 2.18 Hasil pengujian untuk citra 8 dengan pendekatan *deep learning*



Gambar 2.19 Hasil pengujian untuk citra 9 dengan pendekatan *deep learning*



Gambar 2.20 Hasil pengujian untuk citra 10 dengan pendekatan *deep learning*

3. Analisis

Model yang digunakan untuk mendekripsi jenis kendaraan dalam citra adalah OWL-ViT, yaitu model jaringan saraf yang dikembangkan oleh Matthias Minderer dkk. pada tahun 2022. Pengujian menunjukkan kinerja yang relatif baik, di mana pengklasifikasian kendaraan cukup akurat dan *bounding box* yang lebih sempit dibandingkan dengan hasil pendekatan konvensional. Namun, karena model tidak di-*fine-tune* menggunakan data latih spesifik, hasil pengklasifikasian model tidak sempurna, yang ditandai dengan banyaknya mendekripsi berulang terhadap objek yang sama dengan nilai *confidence* yang rendah.

IV. Kesimpulan

AI untuk mendekripsi jenis kendaraan dalam citra berhasil dibangun dengan pendekatan konvensional maupun pendekatan modern. Pendekatan konvensional memanfaatkan algoritma-algoritma pembelajaran mesin tradisional seperti SVM. Algoritma lainnya adalah kNN (*k-Nearest Neighbors*) dan *decision tree*, namun tidak diimplementasikan pada eksperimen kali ini. Fitur citra uji diekstraksi kemudian dibandingkan dengan fitur-fitur dari citra latih dalam suatu *latent space*.

Berbeda halnya dengan pendekatan modern yang memanfaatkan *deep learning* menggunakan arsitektur jaringan saraf tiruan seperti CNN (*Convolutional Neural Network*). Hasil dari metode modern terbukti lebih baik daripada metode konvensional. Bukan hanya karena dapat mendeteksi lebih dari satu kendaraan dalam citra, *bounding box* yang digambarkan juga jauh lebih akurat secara penempatan. Model CNN yang telah di-*fine-tune* ini pun lebih banyak benar dalam memprediksi jenis kendaraan.

V. Komentar dan Refleksi

Tugas kali ini sangat menantang dan tentunya bersinggungan dengan bidang AI. Berkat tugas ini, kami jadi lebih paham bagaimana pendekripsi objek oleh komputer dilakukan, agar komputer dapat mengambil keputusan selanjutnya.

Apabila kami diberikan kesempatan untuk mengembangkan tugas/penelitian ini, kami akan mencoba menggunakan beberapa kombinasi metode yang mungkin. Misalnya untuk metode konvensional, bisa digunakan SVM, *clustering*, dsb. Segmentasi juga bisa menggunakan metode lain seperti *region growing* atau *edge-based*. Untuk metode modern, bisa menggunakan *fine-tuning* terhadap model *pre-trained* CNN lain atau model ANN lainnya.

VI. Alamat GitHub

Kode program dapat diakses pada pranala berikut

<https://github.com/JerichoFletcher/Tugas4-IF4073>