Mid Term Task
Komputasi Intelegensia

Author
Kirono Dwi Saputro
2106656365

Abstract

YOLO (You Only Look Once) adalah algoritma deteksi objek berbasis deep learning yang banyak digunakan dalam bidang visi komputer karena keunggulannya dalam kecepatan dan akurasi deteksi waktu nyata. Penelitian ini berfokus pada penerapan YOLO dalam berbagai konteks komputasi cerdas, seperti pengenalan objek otomatis, analisis video, dan sistem pemantauan. Dengan pendekatan satu langkah pemrosesan gambar, YOLO mampu mengidentifikasi serta mengklasifikasikan objek secara efisien dalam satu tahap, menjadikannya solusi ideal untuk aplikasi real-time yang membutuhkan pemrosesan cepat. Studi ini mengevaluasi performa YOLO versi terbaru—YOLOv4 dan YOLOv5—dalam mendeteksi objek pada beberapa jenis dataset, termasuk COCO dan VOC, yang beragam dalam kompleksitas serta kondisi pencahayaan. Hasil menunjukkan bahwa YOLO menawarkan keseimbangan optimal antara akurasi dan kecepatan pemrosesan, dengan kemampuan adaptasi yang baik terhadap objek yang tumpang tindih dan lingkungan yang kurang ideal. Dengan temuan ini, penelitian ini menyoroti potensi YOLO sebagai alat penting dalam komputasi inteligensia, menawarkan solusi efektif untuk implementasi dalam berbagai aplikasi, termasuk pengawasan keamanan, deteksi anomali, dan sistem otomasi industri.

1 Introduction

Perkembangan pesat dalam deteksi objek telah membawa algoritma YOLO (You Only Look Once) menjadi salah satu pendekatan yang dominan, terutama dalam aplikasi real-time pada bidang komputasi intelijen. YOLO pertama kali diperkenalkan sebagai model deteksi objek berbasis jaringan saraf tiruan yang mampu melakukan identifikasi dan klasifikasi objek secara efisien dan cepat dalam satu kali pemrosesan. Sifat ini memungkinkan YOLO untuk bekerja optimal dalam aplikasi yang memerlukan pemrosesan cepat seperti pengawasan visual, pengendalian kendaraan otonom, serta pengenalan objek dalam robotika.

Penelitian ini bertujuan untuk menguji performa YOLO dalam skenario spesifik menggunakan dataset tertentu untuk mengevaluasi tingkat akurasi, kecepatan pemrosesan, serta ketahanan algoritma dalam berbagai kondisi pencahayaan dan sudut pandang. Eksperimen ini menggunakan beberapa varian YOLO, seperti YOLOv3 dan YOLOv5, yang akan dibandingkan untuk mengidenti-

fikasi versi yang paling optimal dalam kondisi uji yang berbeda. Dengan melakukan evaluasi empiris ini, diharapkan dapat diperoleh wawasan yang lebih mendalam tentang kelebihan dan keterbatasan YOLO dalam berbagai kasus aplikasi, serta memberi kontribusi pada pengembangan aplikasi berbasis deteksi objek yang lebih efektif.

2 Literature Review

2.1 Deteksi Objek dalam Visi Komputer

Deteksi objek merupakan salah satu tugas fundamental dalam visi komputer yang bertujuan untuk mengidentifikasi dan mengklasifikasikan objek dalam citra atau video. Metode tradisional sering kali melibatkan teknik berbasis fitur, seperti Histogram of Oriented Gradients (HOG) dan Scale-Invariant Feature Transform (SIFT). Namun, dengan kemajuan jaringan saraf konvolusional (CNN), pendekatan berbasis pembelajaran mendalam (deep learning) telah menjadi lebih dominan. Berbagai metode seperti R-CNN, Fast R-CNN, dan Faster R-CNN telah diperkenalkan untuk meningkatkan akurasi deteksi, tetapi model-model ini sering kali memiliki kekurangan dalam hal kecepatan dan efisiensi pemrosesan.

2.2 YOLO: You Only Look Once

YOLO, yang diperkenalkan oleh Joseph Redmon dkk. pada tahun 2016, mendefinisikan kembali deteksi objek dengan pendekatan yang revolusioner. Berbeda dengan metode sebelumnya yang menggunakan pendekatan berbasis proposal wilayah, YOLO mengimplementasikan pendekatan single-pass yang memungkinkan deteksi objek dilakukan secara simultan. YOLO membagi gambar input menjadi grid dan menggunakan satu jaringan saraf konvolusional (CNN) untuk memproses gambar secara global, membuat prediksi untuk kotak pembatas dan klasifikasi untuk setiap grid. Dengan melatih pada seluruh gambar, YOLO mengintegrasikan konteks, yang secara signifikan mengurangi kesalahan latar belakang dibandingkan dengan model berbasis wilayah seperti Fast R-CNN.

Pendekatan ini menyederhanakan pipeline deteksi, memungkinkan pelatihan end-to-end dan mencapai kecepatan pemrosesan waktu nyata hingga 45 bingkai per detik pada GPU standar, sementara versi yang lebih cepat, Fast YOLO, dapat memproses 155 bingkai per detik. Hal ini menghasilkan kecepatan deteksi yang sangat cepat, memungkinkan aplikasi real-time di berbagai domain, termasuk pengawasan video, kendaraan otonom, dan interaksi manusia-komputer.

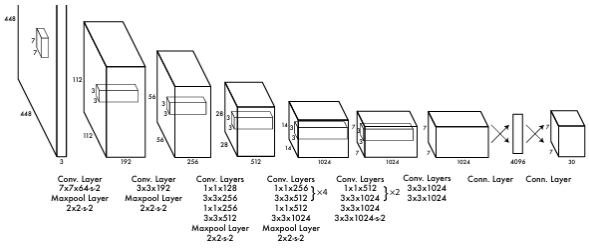

Figure 1: Arsitektur. Jaringan deteksi yang di rancang terdiri dari 24 lapisan konvolusi diikuti oleh 2 lapisan fully connected. dengan menggunakan lapisan konvolusi 1x1 secara bergantian untuk mengurangi dimensi ruang fitur dari lapisan sebelumnya.

2.3 YOLO Version

Sejak diperkenalkan, YOLO telah mengalami beberapa iterasi, termasuk YOLOv2, YOLOv3, dan YOLOv4, dengan masing-masing versi menawarkan peningkatan dalam akurasi dan kecepatan. YOLOv3, misalnya, memperkenalkan penggunaan multi-scale predictions dan residual connections, yang meningkatkan kemampuan model dalam mendeteksi objek kecil. Sementara itu, YOLOv4 yang diluncurkan oleh Bochkovskiy dkk. pada tahun 2020 mengintegrasikan berbagai teknik optimisasi, seperti Mish activation, Cross Stage Partial connections, dan batch normalization, yang meningkatkan performa deteksi tanpa mengorbankan kecepatan.

Baru-baru ini, YOLOv5 diperkenalkan oleh Ultralytics dengan fokus pada kemudahan penggunaan dan performa. YOLOv5 memiliki berbagai ukuran model (small, medium, large, dan xlarge), memungkinkan pengguna untuk memilih model yang paling sesuai dengan kebutuhan aplikasi mereka. Selain itu, YOLOv5 juga dilengkapi dengan kemampuan augmentasi data dan pelatihan yang lebih efisien.

3 Methodology

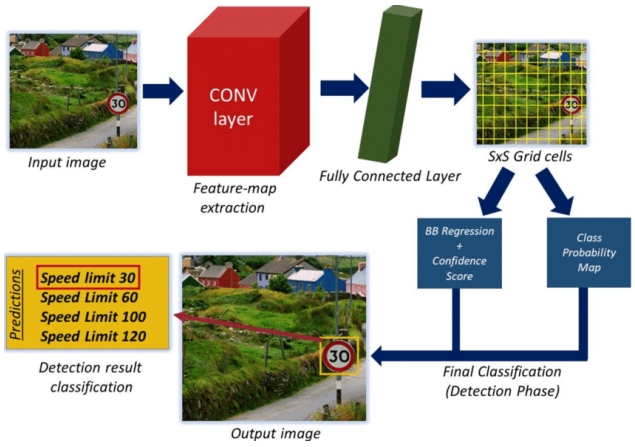

Figure 2: Methods

3.1 Dataset

Langkah pertama adalah memilih file video yang akan dianalisis. Video yang digunakan dalam contoh ini disimpan di Google Drive, yang diakses melalui Google Colab. Path ini mengarah pada file video yang akan diproses. Untuk kompatibilitas dengan Colab, pastikan file video telah terhubung dan dapat diakses.

3.2 Pre-Processing

Praproses melibatkan pemuatan file video dan pengaturan konfigurasi output. Kami menggunakan OpenCV untuk menangani frame video dan menyimpan video yang telah diproses dengan anotasi.

Penjelasan step by step

1. Import Library

   - Kami mengimpor library seperti torch untuk memuat model, cv2 untuk pemrosesan video, dan Video dari IPython.display untuk menampilkan hasil video akhir.

```
import torch
import cv2
from IPython.display import display, Video
```

Figure 3: Import Library

2. Pengaturan Device untuk Optimasi

   - Untuk mengoptimalkan pemrosesan, kami menggunakan GPU jika tersedia; jika tidak, kami akan menggunakan CPU.

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
```

Figure 4: Optimize

3. Membuat Model YOLOv5

   - Kami memuat model YOLOv5 menggunakan Torch Hub. YOLOv5 memiliki beberapa versi (yolov5s, yolov5m, dll), dan yolov5s adalah model yang lebih kecil dan lebih cepat.

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True).to(device)
```

Figure 5: Model YOLO

3.3 Pemrosesan Video dan Inference Model

Setelah memuat model dan mengatur device, kami memproses setiap frame dalam video dan menerapkan model YOLOv5 untuk mendeteksi objek.

1. Memuat File Video

   - Kami memuat file video menggunakan cv2.VideoCapture, yang memungkinkan kami membaca frame satu per satu.

```
video_path = '/content/drive/MyDrive/Data Eki/MVI_6943.MP4'
cap = cv2.VideoCapture(video_path)
```

Figure 6: File

2. Mendapatkan Properti Video

   - Kami mendapatkan properti seperti frame per detik (FPS), lebar, dan tinggi video, yang digunakan untuk membuat output video dengan pengaturan yang sama.

```
fps = int(cap.get(cv2.CAP_PROP_FPS))
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

Figure 7: Properti

3. Mendefinisikan Output Video Writer

   - Kami menentukan codec (mp4v) dan path output untuk video yang telah diproses.

```
# Define the codec and create VideoWriter object to save processed video
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec for .mp4
output_video_path = '/content/drive/MyDrive/Data Eki/output_video.mp4'  # Output path for the processed video
out = cv2.VideoWriter(output_video_path, fourcc, fps, (width, height))
```

Figure 8: Output Name

4. Memproses Frame Video

   - Kami melakukan looping untuk setiap frame dalam video, menerapkan model untuk mendeteksi objek, dan menulis frame yang telah dianotasi ke file output.

```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Melakukan inference pada frame
    results = model(frame)

    # Menampilkan kotak pembatas pada frame
    annotated_frame = results.render()[0]

    # Menulis frame yang telah dianotasi ke output video
    out.write(annotated_frame)
```

Figure 9: Memproses per-Frame

5. Melepas Resource

   - Setelah pemrosesan selesai, kami melepaskan resource untuk mengosongkan memori.

```
cap.release()
out.release()
cv2.destroyAllWindows()
```

Figure 10: Resource

4 Post-processing dan Hasil

Setelah pemrosesan video selesai, video outputnya terletak pada output video path. atau seperti pada gambar dibawah.

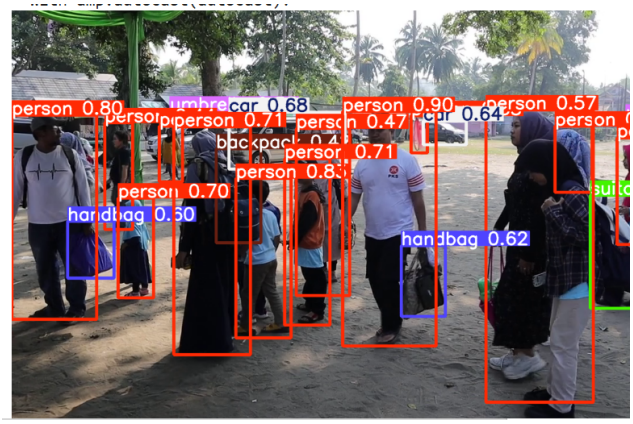

Figure 11: Hasil

5 Kesimpulan

Pada proyek ini, kami mendemonstrasikan bagaimana melakukan deteksi objek pada video menggunakan model YOLOv5. Dengan mengikuti pendekatan ini, kami dapat secara efisien mengan-otasi video frame demi frame, yang berguna untuk berbagai aplikasi seperti pengawasan, kendaraan otonom, dan lainnya.

Reference

- Wang, C. Y., Mark, L., Bochkovskiy, A. (2021). Scaled-YOLOv4: Scaling Cross Stage Partial Network. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. arXiv. https://arxiv.org/pdf/1506.02640
- Liu, J., Li, X., Zhao, X. (2021). YOLOX: Exceeding YOLO Series in Detection Accuracy. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021.
- Jocher, G., et al. (2020). YOLOv5 by Ultralytics. GitHub repository: https://github.com/ultralytics/yolov5.

Lampiran

Google Colab