BAB 2
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian tentang deteksi monyet berbasis image processing telah dilakukan oleh beberapa peneliti. Pertama berjudul "YOLO-SAG: An Improved YOLOv8-Based Model for Wildlife Detection" oleh Lingli Chen, Gang Li, Shunkai Zhang, Wenjie Mao, dan Mei Zhang membahas penerapan model YOLOv8 dalam mendeteksi objek satwa liar pada berbagai kondisi lingkungan. Dalam penelitian tersebut digunakan dataset sebanyak 2083 gambar yang mencakup variasi kondisi objek, seperti occlusion, multi-object, serta perbedaan skala objek. Hasil penelitian menunjukkan bahwa penggunaan dataset dengan jumlah sekitar 2000 gambar yang didukung oleh keberagaman data dan teknik augmentasi mampu menghasilkan performa deteksi yang baik dan stabil. Model YOLO-SAG yang dikembangkan memperoleh nilai *precision* sebesar 94,9%, *recall* sebesar 90,9%, mAP@0.5 sebesar 96,8%, dan mAP@0.5–0.95 sebesar 79,9%. Selain itu, model juga memiliki waktu inferensi sekitar 1,2 ms sehingga mampu melakukan deteksi objek secara real-time dengan baik. Temuan ini menunjukkan bahwa dataset dalam skala kecil hingga menengah tetap dapat memberikan hasil optimal, sehingga relevan dengan pendekatan yang digunakan dalam penelitian ini. (Chen et al., 2024)

Selanjutnya, terdapat jurnal yang berjudul "*Comparing YOLOv8 and Mask R-CNN for instance segmentation in complex orchard environments*" yang ditulis oleh Ranjan Sapkota, Dawood Ahmed, dan Manoj Karkee. Mereka melakukan perbandingan performa YOLOv8 dan Mask R-CNN dalam segmentasi objek pada lingkungan kebun yang kompleks. Hasil penelitian tersebut menunjukkan bahwa penggunaan *confidence threshold* sebesar 0,5 mampu menghasilkan nilai *precision* yang tinggi serta *recall* yang mendekati sempurna, sehingga memberikan keseimbangan yang optimal dalam proses deteksi objek. Pada pengujiannya, model YOLOv8 menghasilkan performa yang lebih baik dibandingkan *Mask R-CNN* dengan nilai *precision* mencapai 0,91 dan *recall* sebesar 0,99 pada *threshold* 0,5. Hasil tersebut menunjukkan bahwa nilai *threshold* 0,5 efektif dalam menyaring prediksi dengan tingkat kesalahan yang rendah tanpa mengurangi jumlah objek

yang berhasil terdeteksi. Temuan ini menunjukkan bahwa nilai *threshold* 0,5 merupakan parameter yang efektif dalam menyaring prediksi model agar tetap akurat tanpa mengurangi jumlah objek yang berhasil terdeteksi. Oleh karena itu, penelitian ini dijadikan sebagai acuan dalam menentukan nilai *threshold* pada sistem yang dikembangkan, karena telah terbukti mampu meningkatkan performa deteksi secara keseluruhan. (Sapkota et al., 2024)

Selanjutnya, terdapat jurnal yang berjudul "*Efficiency-optimized monkey detection for wildlife monitoring: A comprehensive YOLOv8s evaluation*" yang ditulis oleh Rajashekar Kondle dan George Helon Gongaty. Penelitian ini mengembangkan sebuah sistem pendeteksian monyet otomatis menggunakan arsitektur YOLOv8s yang dirancang untuk pemantauan satwa liar secara real-time pada lingkungan dengan keterbatasan sumber daya komputasi. Sistem ini bekerja dengan mengolah gambar digital hasil tangkapan kamera menggunakan model YOLOv8s untuk mendeteksi keberadaan monyet pada berbagai kondisi lingkungan, termasuk kawasan perkotaan, kanopi hutan, dan variasi pencahayaan. Proses pelatihan model dilakukan menggunakan 2.244 data gambar beranotasi yang telah melalui tahapan pra-pemrosesan serta agumentasi data guna meningkatkan kinerja dan ketahanan sistem terhadap perubahan lingkungan. Hasil pengujian menunjukkan bahwa sistem mampu mencapai nilai mAP@0,5 akurasi sebesar 0,52, dengan tingkat presisi 0,89 dan waktu inferensi 5,5 ms, sehingga sistem dinilai memiliki kinerja yang efektif dan efisien untuk diimplementasikan pada perangkat *edge computing* dalam mendukung pemantauan satwa liar otomatis. (Kondle & Gongaty, 2025)

Penelitian selanjutnya yang berjudul "Deteksi Objek Manusia pada Robot Humanoid Menggunakan YOLOv11" yang ditulis oleh M. Azkan Nawal Addufairi dan Gembong Edhi Setyawan. Walaupun penelitian ini fokus pada deteksi manusia, penggunaan algoritma YOLOv11 punya banyak keunggulan karena arsitekturnya ringan dan kemampuan ekstraksi fiturnya efisien, sehingga cocok dipakai di perangkat embedded. Sistem ini diimplementasikan pada robot *humanoid* BRONE yang menggunakan unit komputasi NVIDIA Jetson Orin Nano. Hasilnya, performa deteksinya sangat tinggi dengan nilai presisi 0,995 dan tetap stabil untuk jarak deteksi antara 1 sampai 18 meter. Selain akurat, sistem juga bisa berjalan secara

real-time dengan waktu inferensi rata-rata 26,57 ms dan kecepatan sekitar 27,61 FPS. Bahkan pada kondisi pencahayaan rendah, tingkat keberhasilannya masih mencapai 94,44%. Integrasi sistem deteksi dengan *Robot Operating System* (ROS) memungkinkan koordinat objek yang terdeteksi diubah menjadi sinyal kendali PWM untuk menggerakkan roda *omnidirectional*. Secara teknis, konsep ini juga bisa dikembangkan untuk sistem deteksi dan pelacakan satwa secara otomatis. (Addufairi & Setyawan, 2026)

Penelitian selanjutnya berjudul "Deteksi Jumlah Kendaraan Roda Empat Menggunakan YOLO" yang ditulis oleh Rivaldo Therino, Dedy Hermanto dan Eka Puji Widiyanto. Penelitian ini mengembangkan perangkat lunak untuk menghitung jumlah kendaraan roda empat secara otomatis guna membantu mengatasi kemacetan di perkotaan. Metode yang digunakan adalah algoritma YOLOv8 yang dilatih menggunakan ribuan gambar kendaraan dari platform Roboflow. Pengembangan aplikasinya sendiri memakai bahasa pemograman Python dengan antarmuka berbasis web (Streamlit) agar bisa diakses dengan mudah melalui laptop maupun HP. Sistem ini terbukti sangat efektif karena mampu mengenali kendaraan jenis mobil, bus, dan truk dengan tingkat akurasi mencapai 93%. Melalui pengujian berbagai skenario pelatihan (*epoch*), model terbaik didapatkan pada putaran ke-100 dengan skor performa (mAP) sebesar 79,9%. Selain itu, perangkat lunak ini sukses diuji secara langsung untuk menentukan tingkat kepadatan lalu lintas di jalan raya dengan kecepatan deteksi yang lancar antara 5 hingga 20 frame per detik. (Therino Elevan et al., 2025)

Penelitian selanjutnya yang berjudul "*Preventing Monkey Menace using YOLO based Object Detection Model*" yang ditulis oleh P. Rohit Reddy, M. Vinay Kumar, K. H. Vijaya Kumari, T. Prathima dan Sugamya Katta. Penelitian ini membahas pengembangan sistem deteksi monyet secara otomatis menggunakan metode *deep learning* berbasis YOLOv5 untuk mengatasi gangguan monyet di area pemukiman. Metode yang digunakan dimulai dari pengumpulan dataset yang berasal dari Kaggle dan tambahan gambar dari video Youtube, kemudian dilakukan tahap pre-processing seperti *resize*, *grayscale*, dan pengurangan *noise* untuk meningkatkan kualitas gambar. Data tersebut akan dianotasi menggunakan Roboflow dengan metode *bounding box*, lalu dibagi menjadi data latih, validasi dan

uji. Model YOLOv5 dilatih untuk mendeteksi keberadaan monyet secara real-time dengan cara memprediksi lokasi (bounding box) dan kelas objek dalam gambar. Sistem ini juga dilengkapi dengan mekanisme alert berupa alarm dan pengiriman pesan menggunakan layanan Twilio, serta fitur filtering untuk meminimalkan kesalahan deteksi. Hasil penelitian menunjukkan bahwa sistem mampu mendeteksi monyet secara real-time dengan performa yang baik, ditunjukkan oleh nilai confidence sekitar 0,54 pada contoh deteksi dan nilai AUC pada kurva precision-recall sekitar 0,9. Selain itu, pada berbagai nilai threshold, sistem menghasilkan presisi hingga 0,98 dan recall hingga 0,96, sehingga dapat disimpulkan bahwa metode ini cukup efektif dan berpotensi digunakan untuk membantu mengurangi konflik antara manusia dan monyet di lingkungan pemukiman. (Reddy et al., 2023)

Untuk memperjelas penelitian-penelitian terdahulu yang telah dijelaskan sebelumnya, dapat dilihat dalam tabel dibawah:

Tabel 2.1 Penelitian terdahulu

No	Penulis	Metode	Objek	Hasil
1	(Chen et al., 2024)	Menggunakan metode eksperimen (experimental research) dengan pendekatan deep learning berbasis object detection menggunakan YOLOv8n yang kemudian dimodifikasi menjadi model YOLO-SAG.	Deteksi Satwa liar (*golden snub-nosed monkey*, *Yunnan snub-nosed monkey*, *takin*, *black bear*, *leopard cat*, *wild boar*)	Hasil pengujian menunjukkan bahwa model YOLO-SAG mampu menghasilkan performa deteksi objek satwa liar yang baik, termasuk pada objek monyet. Model tersebut memperoleh nilai Precision sebesar 94,9%, Recall 90,9%, mAP@0.5 sebesar 96,8%, serta mAP@0.5–0.95 sebesar 79,9%. Selain itu, model juga memiliki waktu inferensi yang cepat sekitar 1,2 ms sehingga efektif digunakan untuk deteksi objek secara real-time.

2	(Sapkota et al., 2024)	Metode eksperimen (*experimental research*) dengan pendekatan *deep learning* berbasis *computer vision* untuk membandingkan performa model YOLOv8 dan Mask R-CNN	Deteksi buah stroberi	Hasil pengujian menunjukkan bahwa model YOLOv8 mampu memberikan performa deteksi dan segmentasi objek yang lebih baik dibandingkan Mask R-CNN pada lingkungan kebun yang kompleks. Pada penggunaan confidence threshold sebesar 0,5, model YOLOv8 memperoleh nilai precision sebesar 0,91 dan recall sebesar 0,99, yang menunjukkan bahwa model mampu mendeteksi objek dengan tingkat akurasi tinggi serta meminimalkan kesalahan deteksi. Selain itu, penggunaan threshold 0,5 dinilai efektif dalam menjaga keseimbangan antara jumlah objek yang terdeteksi dan tingkat ketepatan prediksi model.
3	(Kondle & Gongaty, 2025)	Menggunakan varian YOLO v8s (Small) yang dipilih karena keseimbangan terbaik antara akurasi (mAP) dan kecepetan inferensi untuk perangkat edge	Deteksi Monyet	Hasil pengujian menunjukkan bahwa sistem memperoleh nilai akurasi sebesar 0,52 dan presisi sebesar 0,89, dengan waktu inferensi 5,5 ms. Pengujian tersebut dilakukan menggunakan kondisi dunia nyata (real-time),

				bukan hanya menggunakan data dari dataset yang digunakan pada tahap pelatihan model.
4	(Addufairi & Setyawan, 2026)	Menggunakan YOLO v11 varian nano (yolov11n) yang merupakan versi paling ringan.	Deteksi Manusia	Hasil pengujian menunjukkan sistem memperoleh akurasi sebesar 0,995 dan presisi sebesar 0,995 dengan waktu inferensi *real-time* sebesar 26,57 ms. Dataset yang digunakan dikumpulkan menggunakan kamera Logitech C920e di lingkungan kampus dengan variasi jarak dan kondisi pencahayaan guna mempresentasikan kondisi robot yang sesungguhnya dan integrasi sistem robot berhasil menjalankan seluruh skenario - *human-following* dengan tingkat keberhasilan pada pencahayaan terang yaitu 100% dan pada pencahayaan rendah yaitu 94,44%
5	(Therino Elevan et al., 2025)	Menggunakan YOLOv8 yang menerapkan arsitektur *deep learning* terbaru dengan membagi struktur jaringan menjadi tiga	Deteksi kendaraan roda empat	Hasil pengujian menunjukkan implementasi model YOLOv8 dengan konfigurasi 100 *epoch* memberikan performa terbaik dengan presisi nilai rata-rata mencapai 79,9%. Dalam

		bagian yaitu *backbone*, *neck*, dan *head*.		pengujian klasifikasi objek, tingkat akurasi mencapai 96,1% untuk mobil, 100% untuk bus, dan 95% untuk truk menggunakan dataset Roboflow. Waktu inferensi dengan resolusi 4k yaitu 2 FPS dan resolusi 480p yaitu 9 hingga 20 FPS
6	(Reddy et al., 2023)	Menggunakan YOLOv5 sebagai metode utama untuk deteksi monyet dan CNN sebagai dasar pemrosesan gambar	Deteksi Monyet	Hasil pengujian menunjukkan bahwa sistem deteksi monyet menggunakan metode YOLOv5 dapat bekerja dengan baik dalam mengenali monyet pada gambar maupun video secara langsung (real-time). Model ini mampu mendeteksi objek dengan cukup akurat, terlihat dari nilai precision yang tinggi (artinya deteksi yang dilakukan sebagian besar benar) dan recall yang juga baik (artinya sebagian besar monyet berhasil terdeteksi). Dari grafik Precision-Recall, diperoleh nilai AUC sekitar 0,9, yang menandakan performa model sudah bagus dan seimbang antara ketepatan dan kelengkapan deteksi. Selain itu, sistem juga dapat langsung memberikan peringatan

				berupa alarm atau pesan ketika monyet terdeteksi, sehingga bisa membantu pengguna untuk segera mengambil tindakan. Secara keseluruhan, hasil ini menunjukkan bahwa sistem sudah cukup efektif digunakan di kondisi nyata dengan tingkat kesalahan yang rendah.

2.2 Landasan Teori

2.2.1 Dataset

Dataset merupakan komponen utama dalam proses pelatihan model deteksi objek berbasis YOLO. Kualitas dan kuantitas dataset sangat berpengaruh terhadap kemampuan model dalam mengenali objek secara akurat pada berbagai kondisi lingkungan. Dalam konteks deteksi satwa liar, dataset yang digunakan umumnya memiliki karakteristik kompleks, seperti variasi pose objek, kondisi pencahayaan, latar belakang alami, serta adanya objek yang tertutup sebagian (*occlusion*) maupun berukuran kecil (*small object*).

Penelitian yang dilakukan dalam YOLO-SAG: *An Improved YOLOv8-Based Model for Wildlife Detection* menunjukkan bahwa penggunaan dataset dengan jumlah sekitar 2000 gambar sudah cukup untuk melatih model YOLO agar mampu melakukan deteksi objek secara optimal. Dalam penelitian tersebut, digunakan sebanyak 2083 gambar dataset satwa liar yang mencakup berbagai kondisi lingkungan dan variasi objek, termasuk objek yang mengalami *occlusion*, *multi-object*, serta perubahan skala objek. Hasil penelitian menunjukkan bahwa model yang dilatih dengan jumlah dataset tersebut mampu menghasilkan performa deteksi yang baik dan stabil. (Chen et al., 2024)

Selain jumlah dataset, keberagaman data juga menjadi faktor penting dalam meningkatkan kemampuan generalisasi model. Dataset yang mencakup berbagai kondisi seperti objek terlihat secara utuh (*full body*), sebagian tertutup (*occlusion*),

berukuran kecil *(small object*), serta terdapat lebih dari satu objek dalam satu gambar (*multi-object*) dapat membantu model dalam memahami variasi karakteristik objek yang kompleks. Dengan demikian, model tidak hanya belajar dari satu kondisi tertentu, tetapi mampu beradaptasi terhadap berbagai situasi nyata di lapangan.

Lebih lanjut, penelitian tersebut juga menekankan pentingnya teknik augmentasi data untuk meningkatkan variasi dataset tanpa harus menambah jumlah data secara signifikan. Teknik ini dapat membantu model dalam mengenali objek pada kondisi yang belum pernah dilihat sebelumnya, sehingga meningkatkan performa deteksi secara keseluruhan.

Berdasarkan hal tersebut, penggunaan dataset dengan jumlah sekitar 2000 gambar yang disertai dengan variasi kondisi objek dan didukung oleh teknik augmentasi data dinilai cukup representatif untuk melatih model deteksi objek berbasis YOLO agar mampu melakukan generalisasi dengan baik.

2.2.2 Evaluasi Metrik

Evaluasi metrik digunakan untuk mengetahui performa model berdasarkan hasil prediksi terhadap data pengujian. Metrik yang digunakan meliputi *accuracy*, *precision*, *recall*, dan *F1-score*. Setiap metrik memberikan sudut pandang berbeda dalam menilai ketepatan dan kemampuan model dalam mendeteksi objek. (Marcelleno et al., 2025)

Confusion matrix biasanya mencakup empat komponen mendasar, yaitu:

1. *True Positive* (TP): Total dari data yang posistif dan terklasifikasi oleh sistem dengan benar.
2. *True Negative* (TN): Total dari data yang posistif dan terklasifikasi oleh sistem dengan salah.
3. *False Positive* (FP): Total dari data yang negatif dan terklasifikasi oleh sistem dengan benar
4. *False Negative* (FN): Total dari data yang negatif dan terklasifikasi oleh sistem dengan salah.

Confusion matrix ini dapat digunakan untuk menghitung berbagai metrik kinerja seperti *accuracy*, *precision*, *recall*, dan *F1 score*.

Accuracy adalah metrik evaluasi yang digunakan untuk mengukur tingkat kedekatan antara nilai aktual dengan hasil prediksi yang dihasilkan oleh model. Akurasi menunjukkan perbandingan antara jumlah prediksi yang benar dengan seluruh jumlah data yang diuji, sehingga semakin tinggi nilai akurasi, semakin baik kinerja metode atau model tersebut dalam memberikan prediksi yang tepat. Dalam konteks klasifikasi, akurasi mencakup prediksi benar baik pada kelas positif maupun negatif. (Hayati et al., 2023)

Adapun rumus untuk menghitung akurasi sebagai berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision adalah metrik evaluasi yang digunakan untuk mengukur tingkat ketepatan model dalam memprediksi data positif, yaitu seberapa banyak prediksi positif yang benar-benar sesuai dengan kondisi sebenarnya. Dengan kata lain, precision menunjukkan perbandingan antara jumlah data yang diprediksi positif dengan benar (*True Positive*/TP) terhadap seluruh data yang diprediksi sebagai positif (*True Positive + False Positive*). Nilai *precision* yang tinggi menandakan bahwa model memiliki kesalahan yang kecil dalam memberikan prediksi positif, sehingga hasil prediksi lebih dapat dipercaya. (Hayati et al., 2023)

Adapun rumus untuk menghitung presisi sebagai berikut:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall adalah metrik evaluasi yang digunakan untuk mengukur tingkat keberhasilan sistem dalam menemukan seluruh data yang benar-benar positif dari keseluruhan data aktual. Dengan kata lain, *recall* menunjukkan seberapa banyak data positif yang berhasil dideteksi oleh model dibandingkan dengan seluruh data positif yang sebenarnya ada (*True Positive* dan *False Negative*). Nilai *recall* yang tinggi menandakan bahwa model mampu menangkap sebagian besar objek atau informasi yang dicari, sehingga kecil kemungkinan data penting terlewatkan. (Hayati et al., 2023)

Adapun rumus untuk menghitung *recall* sebagai berikut:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1 *Score* adalah metrik evaluasi yang digunakan untuk mengukur keseimbangan antara *precision* dan *recall* dalam suatu model. Nilai F1 *score* diperoleh dari rata-rata harmonis antara *precision* dan *recall*, sehingga memberikan gambaran yang lebih seimbang terhadap performa model, terutama ketika terdapat ketidakseimbangan data. F1 *score* akan tinggi jika nilai *precision* dan *recall* sama sama tinggi, sehingga metrik ini sangat berguna untuk memastikan bahwa model tidak hanya akurat dalam prediksi positif, tetapi juga mampu menangkap sebagian besar data yang relevan. (Hayati et al., 2023)

Adapun rumus untuk menghitung F1 *score* sebagai berikut:

$$F1\ Score = 2x\ \frac{precision\ x\ recall}{precision + recall} \quad (4)$$

2.2.3 Webcam

Webcam (*web camera*) merupakan perangkat kamera digital yang terhubung dengan komputer atau laptop dan berfungsi untuk mengambil gambar maupun video secara langsung (*real-time*). Perangkat ini umumnya ditempatkan pada bagian atas monitor atau telah terintegrasi pada laptop. Selain kamera, beberapa jenis webcam juga dilengkapi dengan mikrofon, sensor, dan lampu indikator sebagai penanda bahwa perangkat sedang digunakan. Cara kerja webcam dimulai dari proses penangkapan cahaya oleh lensa, kemudian informasi tersebut dikonversi menjadi data digital yang dapat ditampilkan pada layar maupun dikirimkan melalui jaringan internet. Dalam penggunaannya, webcam banyak dimanfaatkan untuk berbagai aktivitas, seperti rapat daring, pembelajaran jarak jauh, *live streaming*, dan dokumentasi video. Oleh karena itu, webcam menjadi salah satu perangkat yang mendukung komunikasi digital karena memungkinkan penyampaian informasi secara visual tanpa terbatas oleh jarak. (Edavos, n.d.)

2.2.4 Pengenalan Python

Python merupakan bahasa pemrograman tingkat tinggi yang dirancang dengan sintaks sederhana sehingga mudah dibaca dan dipahami, karena struktur perintahnya menyerupai bahasa manusia. Bahasa pemrograman ini dikembangkan oleh Guido van Rossum pada akhir tahun 1980 dan hingga kini terus berkembang

serta digunakan secara luas karena bersifat serbaguna, mulai dari pengembangan aplikasi web hingga pengolahan data dan penelitian komputasi yang kompleks. Salah satu keunggulan utama Python adalah efisiensi dalam pengembangan perangkat lunak, di mana pengembang tidak perlu menuliskan seluruh fungsi dari awal karena Python menyediakan berbagai modul dan pustaka siap pakai yang dapat langsung dimanfaatkan. Keberadaan modul-modul tersebut mampu mempercepat proses pengembangan aplikasi serta memudahkan perawatan dan pengembangan sistem di masa mendatang.

Dalam penerapannya, Python menggunakan mekanisme penerjemahan melalui *interpreter* yang mengeksekusi instruksi program secara baris per baris, sehingga memberikan fleksibilitas yang tinggi selama proses pengembangan. Selain itu, Python memiliki kemampuan integrasi yang baik dengan berbagai teknologi lain, seperti pengelolaan basis data secara otomatis melalui kerangka kerja Django serta dukungan terhadap perhitungan matematika dan komputasi yang kompleks. Dengan sifatnya yang bersifat bebas digunakan (*open-source*) dan kompatibel dengan berbagai sistem operasi, Python banyak dimanfaatkan dalam berbagai bidang teknologi modern, antara lain pengembangan kecerdasan buatan, pembuatan antarmuka aplikasi, hingga pengembangan logika pada industri permainan komputer. (Umarjonovich, 2023)

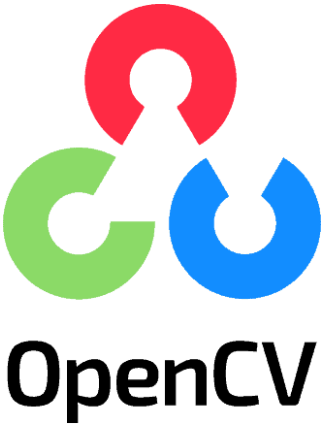
Gambar 2. 1 Python[1]

2.2.5 Pengenalan OpenCV

OpenCV (Open-Source Computer Vision Library) adalah sebuah software library sumber terbuka (*open-source*) yang berfokus pada visi komputer (*computer*

[1] "Python Logo: History, Meaning & Brand", diakses 14 April 2026, https://logos-world.net/python-logo/

vision) dan pembelajaran mesin (*machine learning*). Proyek ini pertama kali diinisiasi pada tahun 1999 oleh Gary Bradsky dan timnya di pusat riset intel, Rusia, dengan rilis pertama tahun 2000. Saat ini, OpenCV dikelola oleh komunitas pengembangan dibawah naungan OpenCV Foundation dan telah menjadi standar industri maupun akademis untuk aplikasi *real-time*. (Dhamodaran et al., 2024)



Gambar 2. 2 OpenCV[2]

Sebagai pustaka yang serbaguna, OpenCV mendukung berbagai bahasa pemrograman populer seperti Python, C, C++, dan Java. Pustaka ini menyediakan lebih dari 2.500 algoritma yang dioptimalkan, mencakup berbagai fungsi mulai dari teknik visi komputer tradisional hingga algoritma tingkat lanjut. Beberapa algoritma utama yang tersedia yaitu:

1. Haar Cascades: Digunakan untuk mendeteksi objek dan wajah.
2. SIFT (*Scale-Invariant Feature Transform*): Untuk identifikasi fitur gambar.
3. ORB (*Oriented FAST and Rotated BRIEF*): Alternatif efisien untuk deskripsi fitur.

OpenCV dibuat untuk membantu proses komputasi menjadi lebih efisien sehingga pengembangan aplikasi computer vision dapat dilakukan dengan lebih cepat dan mudah. Dalam penggunaannya, OpenCV sering digabungkan dengan pustaka pengolahan data seperti NumPy dan Pandas, serta framework kecerdasan buatan seperti TensorFlow dan PyTorch untuk melatih jaringan saraf (*neurol networks*) yang kompleks.

[2] "Official OpenCV Logos & Media Kit", OpenCV, diakses 14 April 2026, https://opencv.org/media-kit/

Secara fungsional, OpenCV memiliki kemampuan luas dalam pemrosesan visual antara lain:

1. Membaca, mengubah ukuran (*resizing*), dan memutar (*rotating*) gambar.
2. Ekstraksi Region of Interest (ROI) dan analisis nilai RGB pada piksel.
3. Deteksi objek, segmentasi gambar, dan pengenalan tulisan tangan.
4. Pemrosesan video dan aliran data kamera secara *real-time*. (Dhamodaran et al., 2024)

2.2.6 Algoritma YOLO (*You Only Look Once*)

YOLO merupakan salah satu model deteksi objek dan segmentasi gambar yang banyak digunakan, yang dikembangkan oleh Joseph Redmon dan Ali Farhadi dari Universitas Washington. Sejak diperkenalkan pada tahun 2015, YOLO dikenal luas karena kemampuannya menghasilkan deteksi dengan kecepatan tinggi tanpa mengorbankan tingkat akurasi. YOLO termasuk ke dalam algoritma deteksi objek state-of-the-art yang memanfaatkan arsitektur Convolutional Neural Network (CNN). Tidak seperti metode deteksi konvensional yang memproses gambar melalui beberapa tahap region proposal, YOLO merumuskan proses deteksi sebagai satu permasalahan regresi terpadu. Dengan pendekatan ini, seluruh tahapan mulai dari ekstraksi fitur, prediksi bounding box, hingga penentuan kelas objek dilakukan secara simultan dalam satu kali proses forward pass pada jaringan saraf.

Secara teknis, proses kerja YOLO dimulai ketika gambar input dimasukkan ke dalam jaringan CNN untuk diproses melalui beberapa *convolution layer*. Pada proses ini, jaringan akan mengekstraksi berbagai fitur penting dari gambar seperti tepi, tekstur, bentuk, dan pola objek. Hasil ekstraksi fitur tersebut kemudian diolah lebih lanjut oleh jaringan untuk menghasilkan prediksi lokasi objek, nilai *confidence*, dan klasifikasi objek dalam satu kali proses inferensi. (Ali, 2024)

Selain itu, secara teknis keluaran (*output*) dari model YOLO berbentuk tensor dengan dimensi:

$$S \times S \times (B \times 5 + C)$$

Dimana S merupakan jumlah *grid cell* yang membagi gambar, B adalah jumlah bounding box yang diprediksi setiap grid, dan C adalah jumlah kelas objek.

Nilai 5 merepresentasikan parameter *bounding box* yang terdiri dari koordinat posisi (x, y), lebar (w), tinggi (h), serta nilai *confidence*. Dengan struktur ini, YOLO mampu melakukan prediksi lokasi dan klasifikasi objek secara bersamaan dalam satu kali proses.

Secara umum, proses kerja YOLO meliputi beberapa tahapan utama, yaitu:

1. *Feature exctraction*: Pada tahap ini CNN digunakan untuk mengekstraksi fitur penting dari gambar seperti tepi, tekstur, dan pola objek. Setiap *convulation layer* melakukan proses konvolusi menggunakan filter tertentu untuk menghasilkan *feature map* yang merepresentasikan karakteristik objek pada gambar.
2. *Grid Cell Division*: Setelah *feature map* diperoleh, gambar akan dibagi menjadi beberapa *grid cell*. Setiap *grid cell* bertanggung jawab untuk mendeteksi objek yang berada pada area tertentu dalam gambar. Masing-masing grid akan memprediksi sejumlah *bounding box* beserta *confidence score* dan probabilitas kelas objek.
3. *Bounding box prediction*: Proses prediksi lokasi objek dilakukan dalam bentuk *bounding box* yang direpresentasikan menggunakan koordinat pusat objek (x, y), lebar (w), dan tinggi (h). Selain itu, model juga menghasilkan nilai *confidence score* yang menunjukkan tingkat keyakinan bahwa *bounding box* tersebut mengandung objek.
4. *Class probality prediction*: Pada tahap ini model menentukan probabilitas kelas objek pada setiap *bounding box* yang telah diprediksi. Probabilitas tertinggi akan digunakan sebagai hasil klasifikasi objek yang terdeteksi.
5. *Non-Maximum Suppresion*: Tahap ini digunakan untuk menghilangkan deteksi ganda pada objek yang sama. Sistem akan memilih *bounding box* dengan nilai *confidence* tertinggi dan menghapus *bounding box* lain yang memiliki tingkat tumpang tindih tinggi berdasarkan nilai *Intersection over Union* (IoU).

Dalam proses pelatihan model YOLO, terdapat beberapa parameter penting yang perlu diatur. Parameter tersebut meliputi *epoch* (jumlah iterasi pelatihan), *batch size* (jumlah data dalam satu iterasi), *learning rate* (kecepatan pembaruan bobot), serta *image size* yang digunakan sebagai input model. Selain itu digunakan

juga *confidence threshold* untuk menentukan batas minimal deteksi objek dan *Intersection over Union* (IoU) untuk mengukur tingkat kesesuaian antara prediksi dan data sebenarnya. Pengaturan paramter ini sangat berpengaruh terhadap peforma model dalam akurasi dan kecepatan deteksi.

Epoch merupakan jumlah perulangan seluruh dataset selama proses pelatihan berlangsung. Semakin besar nilai *epoch*, maka model memiliki lebih banyak kesempatan untuk mempelajari pola dari data. Namun, penggunaan *epoch* yang terlalu besar dapat menyebabkan *overfitting*, yaitu kondisi ketika model terlalu menyesuaikan diri dengan data pelatihan sehingga kurang mampu mengenali data baru.

Batch size adalah jumlah data yang diproses dalam satu kali iterasi sebelum dilakukan pembaruan bobot. Penggunaan *batch size* yang kecil cenderung membuat proses pelatihan lebih ringan namun kurang stabil, sedangkan *batch size* yang besar dapat mempercepat pelatihan dan menghasilkan pembaruan yang lebih stabil, tetapi membutuhkan kapasitas memori yang lebih besar.

Learning rate merupakan parameter yang mengatur seberapa besar perubahan bobot model pada setiap iterasi. Nilai *learning rate* yang terlalu besar dapat menyebabkan proses pelatihan menjadi tidak stabil, sedangkan nilai yang terlalu kecil akan membuat proses pelatihan berlangsung lambat. Oleh karena itu, diperlukan penetuan nilai *learning rate* yang tepat agar model dapat mencapai konvergensi secara optimal.

Image size adalah ukuran resolusi gambar yang digunakan sebagai input model. Ukuran gambar lebih besar memungkinkan model menangkap detail objek dengan lebih baik sehingga dapat meningkatkan akurasi, namun berdampak pada waktu komputasi yang lebih lama. Sebaliknya, ukuran yang lebih kecil akan mempercepat proses pelatihan dan inferensi, tetapi dapat mengurangi detail informasi pada objek.

Confidence threshold merupakan parameter yang digunakan untuk menentukan batas minimum tingkat keyakinan model dalam mendeteksi suatu objek. Nilai confidence diperoleh dari hasil perhitungan model deteksi objek, yang secara umum merupakan kombinasi antara probabilitas keberadaan objek pada suatu bounding box dengan tingkat kesesuaian prediksi terhadap data sebenarnya

yang diukur menggunakan Intersection over Union (IoU). Secara matematis, nilai confidence dapat dinyatakan sebagai hasil perkalian antara probabilitas objek dan nilai IoU. Dalam implementasi sistem, nilai confidence threshold yang digunakan adalah sebesar 0,5.

Pemilihan nilai *confidence threshold* sebesar 0,5 dalam deteksi objek menggunakan YOLO didasarkan pada kemampuannya dalam memberikan keseimbangan yang optimal antara *precision* dan *recall*. Model YOLOv8 yang diuji pada berbagai kondisi menunjukkan bahwa penggunaan *threshold* 0,5 mampu menghasilkan nilai precision yang tinggi serta *recall* yang mendekati sempurna, sehingga model memiliki tingkat keyakinan yang cukup dalam membedakan prediksi yang benar dan salah tanpa mengorbankan banyak deteksi objek yang relevan. Dengan demikian, *threshold* 0,5 dapat dianggap sebagai nilai yang efektif dan umum digunakan dalam menentukan validitas hasil deteksi, karena mampu menjaga keseimbangan performa model secara keseluruhan. (Sapkota et al., 2024)

$$Confidence = P(Object) \times IoU_{pred}^{truth} \quad (5)$$

Intersection over Union (IoU) digunakan untuk mengukur tingkat kesesuaian antara *bounding box* hasil prediksi dengan *bounding box* sebenarnya. Nilai IoU dihitung berdasarkan perbandingan antara area irisan (*overlap*) dan area gabungan (*union*) dari kedua *bounding box* tersebut.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \quad (6)$$

Semakin tinggi nilai IoU, maka semakin baik kesesuaian antara prediksi dan data sebenarnya, sehingga menunjukkan bahwa model memiliki tingkat akurasi deteksi yang tinggi.

Dalam proses pelatihan, YOLO menggunakan fungsi kerugian (*loss function*) yang terdiri dari beberapa komponen utama, yaitu *localization loss* untuk mengukur kesalahan posisi *bounding box*, *confidence loss* untuk mengukur kesalahan dalam mendeteksi keberadaan objek, serta *classification loss* untuk mengukur kesalahan dalam menentukan kelas objek. Kombinasi dari tiga komponen ini memungkinkan model untuk belajar secara optimal dalam mendeteksi lokasi dan jenis objek secara bersamaan.

2.2.7 Roboflow

Roboflow adalah platform lengkap untuk mengembangkan model computer vision yang memudahkan pengguna dalam mengelola seluruh proses, mulai dari pengolahan dataset hingga model siap digunakan. Platform ini menyediakan fitur seperti manajemen dataset, pelabelan data, preprocessing, augmentasi, hingga pelatihan dan deployment model. Dengan antarmuka yang sederhana dan dukungan integrasi ke berbagai framework seperti TensorFlow, PyTorch, dan YOLO, Roboflow dapat digunakan baik oleh pemula maupun pengembang yang sudah berpengalaman untuk membangun sistem deteksi objek secara efisien.

Secara teknis, Roboflow digunakan sebagai alat untuk mengelola dan memproses dataset sebelum data digunakan pada tahap training model deep learning. Dataset yang diunggah ke dalam platform tidak langsung dipakai, tetapi terlebih dahulu melewati beberapa tahap pengolahan agar kualitas data menjadi lebih rapi, konsisten, dan sesuai dengan kebutuhan model computer vision.

Dalam konteks penggunaan YOLO, Roboflow menyediakan opsi ekspor dataset dalam format YOLO. Pada format ini, setiap anotasi objek disimpan dalam bentuk file teks (.txt) dengan format *class x_center y_ center width height*. Seluruh nilai koordinat dinormalisasi dalam rentang 0 hingga 1 berdasarkan ukuran gambar. Sehingga sesuai dengan kebutuhan model YOLO. (Yamani et al., 2025)

Gambar 2. 3 Roboflow[3]

Dalam Roboflow, dataset yang diunggah akan melalui beberapa tahapan pengolahan sebagai berikut:

1. Pelabelan data (*Annotation*): Merupakan proses pemberian tanda objek tertentu dalam gambar, biasanya dalam bentuk *bounding box* atau segmentasi. Proses ini bertujuan untuk memberikan informasi kepada

[3] "Peridio Partners: Roboflow", Peridio, diakses 14 April 2026, https://www.peridio.com/partners/roboflow

model mengenai objek apa yang harus dikenali, sehingga termasuk dalam pendekatan *supervised learning*.

2. Preprocessing Data: Tahap awal pengolahan data untuk meningkatkan kualitas dan konsistensi dataset. Beberapa teknik yang umum dilakukan antara lain *resizing* (penyesuaian ukuran gambar), normalisasi nilai piksel, serta perbaikan orientasi gambar. Tahap ini penting agar data yang digunakan memiliki format yang seragam dan mudah diproses oleh model.
3. Augmentasi Data (*Data Augmentation)*: Merupakan teknik untuk meningkatkan variasi dataset dengan memodifikasi data yang sudah ada, seperti. rotasi, *flipping*, perubahan pencahayaan, *zoom*, dan *cropping*. Tujuan dari augmentasi adalah untuk meningkatkan kemampuan generalisasi model serta mengurangi risiko overfitting.
4. Pembagian Dataset (*Dataset Splitting*): Dataset yang telah di proses kemudian dibagi menjadi data latihan (*training*), validasi (*validation*), dan pengujian (*testing*). Pembagian ini bertujuan untuk memastikan model dapat dievaluasi secara objektif selama dan setelah proses pelatihan.
5. Export Dataset: Setelah seluruh proses selesai, dataset akan diekspor ke dalam format tertentu, seperti format YOLO, sehingga dapat langsung digunakan dalam proses pelatihan model.

Selain tahapan tersebut, terdapat beberapa *settingan* pada Roboflow yang perlu diperhatikan agar dataset yang dihasilkan sesuai dengan kebutuhan model, khususnya pada implementasi YOLO. *Settingan* ini dilakukan pada saat pembuatan *dataset version* dan secara langsung memengaruhi kualitas data yang digunakan dalam proses pelatihan model. Secara teknis, konfigurasi yang digunakan meliputi:

1. *Resize Image*: Untuk menyeragamkan ukuran input gambar (misalnya 416x416 atau 640x640 piksel), yang berpengaruh pada keseimbangan antara akurasi dan kecepatan prose.
2. *Auto-Orient* dan Normalisasi: Untuk memastikan orientasi gambar benar serta menstabilkan distribusi nilai piksel agar proses pelatihan lebih optimal.
3. Augmentasi Data: Seperti *flip*, rotasi, dan perubahan pencahayaan untuk meningkatkan variasi data dan kemampuan generalisasi model,

4. Pembagian Dataset: Membagi dengan rasio umumnya *training* 70%, *validation* 20%, dan *testing* 10% untuk mendukung proses pelatihan dan evaluasi yang objektif.
5. Format Anotasi YOLO: Dengan struktur *class x_center y_center width height* dalam bentuk normalisasi (0-1).
6. *Generate* dan Ekspor Dataset: Untuk menghasilkan dataset akhir dalam format YOLO lengkap dengan struktur folder dan file konfigurasi.

Secara keseluruhan, Roboflow berperan sebagai tahap awal dalam pipeline sistem deteksi objek, dimana dataset yang telah melalui proses pengolahan akan digunakan sebagai input dalam pelatihan model YOLO. Dengan demikian, Roboflow dan YOLO saling terintegrasi dalam satu alur sistem, mulai dari pengolahan data hingga menghasilkan model deteksi objek yang siap digunakan.

2.2.8 Visual Studio Code

Visual Studio Code merupakan aplikasi editor source code editor berbasis open-source yang dikembangkan oleh Microsoft. VS Code dikategorikan sebagai lingkungan pengembangan terintegras (Integrated Development Environment/IDE) yang ringan namun memiliki fungsionalitas yang kuat untuk mendukung berbagai bahasa pemrograman dan teknologi.

VS Code memiliki banyak fitur yang membantu programmer dalam menulis, memperbaiki (debug), dan mengelola kode program. VS Code dapat memberikan saran kode secara otomatis sesuai dengan konteks, sehingga mempercepat proses penulisan dan mengurangi kesalahan. Proses debugging juga bisa dilakukan langsung di dalam editor tanpa perlu membuka aplikasi lain. Selain itu, VS Code mendukung berbagai bahasa pemrograman seperti Python, Java, JavaScript, C++, C#, dan lainnya melalui fitur ekstensi. Pengguna juga dapat menambahkan berbagai ekstensi sesuai kebutuhan untuk menyesuaikan fungsi VS Code. VS Code terintegrasi dengan Git sehingga memudahkan pengelolaan kode dan kerja sama tim. Tampilan antarmukanya pun bersih, sederhana, dan mudah digunakan.

Berdasarkan hasil penelitian, penggunakan VS Code terbukti meningkatkan pemahaman konseptual keterampilan *coding* bagi pemula secara signifikan karena

kemampuannya dalam menyederhanakan proses penulisan, pengetesan (*debugging*) dan manajemen file program. Hal ini menjadikan VS Code sebagai salah satu pilihan teks editor terbaik dalam mendukung implementasi algoritma pemrograman yang kompleks, termasuk dalam pengembangan sistem berbasis Python dan *computer vision*. (Murani et al., 2025)

Gambar 2. 4 Visual Studio Code[4]

VS Code bukan hanya menyediakan fitur-fitur editor sederhana, tetapi juga dilengkapi dengan banyak komponen dan alat yang membantu pengembangan perangkat lunak secara efisien. Penggunaan VS Code mencakup beberapa aspek penting, seperti antarmuka pengguna, pengelolaan proyek, fitur penulisan kode, terminal, kontrol versi, serta kemampuan extensi yang fleksibel. (Microsoft, n.d.)

[4] "Visual Studio Code Brand Assets", Microsoft, diakses 14 April 2026, https://code.visualstudio.com/brand