BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Dalam bagian ini dijelaskan beberapa penelitian yang terkait dengan topik penelitian tentang deteksi pelanggaran penggunaan helm pada pengendara motor. Studi-studi ini dijadikan referensi dan landasan bagi penelitian.

1) Deteksi Pemakaian Helm Proyek Dengan Metode *Convolutional Neural Network*

   Penelitian ini mengembangkan sistem deteksi penggunaan helm proyek oleh pekerja konstruksi menggunakan metode *Convolutional Neural Network (CNN)* dengan algoritma YOLOv2. Sistem ini mencakup tiga tahap utama, yaitu *preprocessing*, *training* dengan *transfer learning*, serta deteksi menggunakan *anchor box*, IOU, dan *Non-Max Suppression* untuk memastikan akurasi *bounding box*. Pengujian sistem menunjukkan bahwa *F1-score* yang diperoleh adalah 0,79, dengan kinerja lebih optimal pada deteksi pekerja individu dibandingkan kelompok, karena pada kondisi berkelompok sering terjadi penutupan tubuh antar pekerja yang memengaruhi akurasi. Sistem ini memberikan solusi untuk membantu pengawasan penggunaan helm proyek di lokasi kerja secara efektif, terutama pada pekerja individu (Widodo dkk., 2021).

2) Deteksi Pengendara Motor Tanpa Menggunakan Helm Dengan Algoritma *Deep Learning* YOLO

   Penelitian ini mengembangkan sistem pendeteksi helm pada pengendara motor menggunakan metode *Convolutional Neural Network (CNN)* dengan algoritma YOLO untuk mengidentifikasi pengendara yang menggunakan helm atau tidak. Sistem ini mampu mendeteksi pelanggaran dengan akurasi rata-rata sebesar 89,04% dan nilai rata-rata *training loss* sebesar 1,2. Selain mendeteksi, sistem ini juga dilengkapi fitur *Counting People Object* untuk menghitung jumlah pelanggar dan bukan pelanggar. Hasil penelitian menunjukkan kinerja yang baik dalam mendeteksi objek di area kepala, meskipun beberapa kondisi tertentu dapat memengaruhi hasil pendeteksian. Penelitian ini dapat dikembangkan lebih lanjut dengan algoritma YOLO versi

terbaru atau metode lain yang lebih canggih untuk meningkatkan performa (Khoiriyah & Armawan, 2023).

3) Deteksi *Helmet* dan *Vest* Keselamatan Secara *Realtime* Menggunakan Metode YOLO Berbasis *Web Flask*

Penelitian ini mengembangkan sistem deteksi penggunaan helm dan *vest* keselamatan secara *real-time* menggunakan algoritma YOLOv2 berbasis *web Flask*. Sistem ini melalui tahapan pengumpulan data citra, anotasi data, pelatihan model, dan *deployment*, serta berhasil mendeteksi penggunaan helm dan *vest* dengan *bounding box* berwarna hijau, dan pelanggaran dengan *bounding box* merah. Dengan menggunakan *dataset* 1.142 gambar yang dibagi menjadi empat kelas (*helmet, vest, no_helmet, no_vest*), sistem ini mencapai akurasi rata-rata 81,60%, validasi mAP sebesar 76,68%, dan *avg loss* 0,173%. Meskipun hasilnya baik, faktor seperti pencahayaan, posisi objek kamera, dan jarak objek memengaruhi kinerja deteksi. Penelitian ini menunjukkan bahwa tekno*log*i *deep learning*, khususnya YOLOv2, dapat efektif digunakan untuk pengawasan penggunaan APD di lingkungan kerja secara otomatis dan *real-time* (Hatami dkk., 2023).

4) Deteksi Pelanggaran Lalu Lintas Tidak Menggunakan Helm Dengan YOLOv4 Pada Sistem ETLE

Penelitian ini mengembangkan sistem deteksi pelanggaran penggunaan helm pada pengendara sepeda motor dengan menggunakan algoritma YOLOv4 berbasis *deep learning* dan arsitektur *Convolutional Neural Network (CNN)*. Sistem ini memanfaatkan rekaman video dari *IP Camera* yang dipasang pada mobil patroli polisi lalu lintas dan terintegrasi dengan *Electronic Traffic Law Enforcement (E-TLE)* untuk penindakan digital tanpa interaksi langsung. YOLOv4, yang dirilis pada 2020, digunakan untuk mendeteksi objek secara *real-time* dengan akurasi dan kecepatan yang lebih baik dibandingkan versi sebelumnya. *Dataset* yang digunakan berasal dari rekaman video yang dilabeli dan dipisah untuk mengetahui koordinat dan data kendaraan pelanggar. Hasil penelitian menunjukkan bahwa sistem ini berhasil mendeteksi pelanggaran penggunaan helm dengan baik dan dapat diterapkan untuk penindakan berbasis *mobile* (Suryanto & Kardian, 2023).

5) Sistem Deteksi Helm Menggunakan Algoritma YOLOv8

Penelitian ini mengembangkan sistem deteksi penggunaan helm pada pengendara motor berbasis *deep learning* dengan algoritma YOLOv8 yang diimplementasikan secara *real-time* menggunakan Jetson Nano. Sistem ini mampu mendeteksi dan mengidentifikasi pelanggaran penggunaan helm, dengan performa tinggi yang ditunjukkan oleh *f1-score* sebesar 91,1% untuk kelas 'Helm' dan 33,0% untuk kelas 'Tidak Helm'. Hasil penelitian ini mengindikasikan bahwa tekno*log*i berbasis YOLOv8 dapat diandalkan untuk mendeteksi helm pada pengendara motor secara akurat dan efisien (Bintang, 2024).

Tabel 2. 1 Penelitian Terdahulu

No	Nama Peneliti	Judul Penelitian	Algoritma	Hasil
1	(Widodo dkk., 2021)	Deteksi Pemakaian Helm Proyek Dengan Metode *Convolutional Neural Network*	YOLOv2 + CNN	*F1-score*: 0,79
2	(Khoiriyah & Armawan, 2023)	Deteksi Pengendara Motor Tanpa Menggunakan Helm Dengan Algoritma *Deep Learning* YOLO	YOLO + CNN	Akurasi: 89,04%; *Loss*: 1,2
3	(Hatami dkk., 2023)	Deteksi *Helmet* dan *Vest* Keselamatan Secara *Realtime* Menggunakan Metode YOLO Berbasis *Web Flask*	YOLOv2	Akurasi: 81,6%; mAP: 76,68%; *Loss*: 0,173
4	(Suryanto & Kardian, 2023)	Deteksi Pelanggaran Lalu Lintas Tidak Menggunakan Helm Dengan	YOLOv4 + CNN	Berhasil mendeteksi pelanggaran helm secara otomatis.

No	Nama Peneliti	Judul Penelitian	Algoritma	Hasil
		YOLOv4 Pada Sistem ETLE		
5	(Bintang, 2024)	Sistem Deteksi Helm Menggunakan Algoritma YOLOv8	YOLOv8	*F1-score*: 91,1% (Helm); 33% (Tidak Helm)

2.2 Landasan Teori

2.2.1 Penggunaan Helm

Helm merupakan alat pelindung kepala yang dirancang untuk mengurangi risiko cedera akibat benturan saat terjadi kecelakaan. Setiap pengendara dan penumpang sepeda motor wajib menggunakan helm yang memenuhi Standar Nasional Indonesia (SNI). Helm berfungsi untuk melindungi kepala dari benturan langsung dan mengurangi risiko cedera fatal (Kementerian Perhubungan Republik Indonesia, 2019).

Helm yang sesuai dengan standar SNI 1811:2007 untuk pengendara sepeda motor harus memenuhi beberapa kriteria penting. Helm harus terbuat dari material yang kuat dan mampu menyerap benturan, dengan desain yang mencakup bantalan pelindung di dalamnya. Tali dagu (*chin strap*) harus dikencangkan dengan aman, dan helm dapat berupa jenis terbuka atau tertutup (*full face*). Selain itu, helm harus mencantumkan *log*o, merek, dan kode SNI yang di-embos. Helm yang memenuhi standar ini dapat membantu mengurangi risiko cedera pada pengendara (Standar Nasional Indonesia (SNI), 2013).

2.2.2 *Deep Learning*

Deep Learning adalah cabang kecerdasan buatan yang terinspirasi dari cara kerja otak manusia, menggunakan jaringan saraf tiruan (*neural networks*) untuk memproses data. Tekno*log*i ini memungkinkan komputer untuk belajar dan beradaptasi secara mandiri, mengenali berbagai jenis data seperti foto, video, dan teks. Meskipun telah diperkenalkan sejak 1950-an, perkembangan pesat *deep learning*

baru terjadi setelah 40 tahun, didorong oleh kemajuan dalam komputasi dan ketersediaan data besar (Nurhakiki & Yahfizham, 2024).

Mekanisme kerjanya melibatkan beberapa lapisan jaringan saraf yang memproses informasi secara bertingkat, di mana setiap lapisan berfungsi untuk mengekstrak dan mengintegrasikan fitur-fitur kompleks dari data yang tidak terstruktur. Dalam proses pelatihan model *deep learning*, terdapat beberapa konsep penting, seperti *epoch*, *batch size*, *optimizer*, dan *learning rate*.

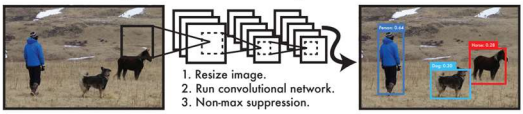
1) *Epoch* adalah satu siklus penuh di mana seluruh data latih diproses oleh model. Sebuah model biasanya dilatih dalam banyak *epoch* untuk memastikan model mampu belajar pola secara menyeluruh dari data.
2) *Batch size* adalah jumlah data yang diproses sekaligus dalam satu iterasi. *Batch size* yang lebih kecil memerlukan lebih banyak iterasi per *epoch* tetapi memungkinkan pelatihan pada perangkat dengan memori terbatas.
3) *Optimizer* adalah algoritma yang digunakan untuk memperbarui bobot model berdasarkan hasil perhitungan *loss function*. Contoh *optimizer* populer adalah *Stochastic Gradient Descent* (SGD) dan *Adam Optimizer*.
4) *Learning rate* adalah parameter yang menentukan seberapa besar perubahan bobot model dalam setiap langkah pembaruan. Nilai *learning rate* yang terlalu besar dapat menyebabkan model gagal menemukan solusi optimal, sedangkan nilai yang terlalu kecil membuat proses pelatihan memakan waktu lebih lama (Goodfellow dkk., 2016).

Dengan kemampuan untuk mengenali pola rumit, *deep learning* telah diterapkan dalam berbagai bidang, termasuk pengenalan wajah, penerjemahan bahasa, sistem rekomendasi, mobil otonom, dan diagnosa medis. Keunggulan tekno*log*i ini terletak pada akurasinya yang tinggi dan kemampuannya untuk menangani data kompleks. Namun, tantangan yang dihadapi meliputi kebutuhan akan data yang besar, komputasi yang intensif, dan kesulitan dalam menginterpretasikan model yang dihasilkan. Secara keseluruhan, *deep learning* merupakan tekno*log*i transformatif yang merevolusi cara komputer memahami dan berinteraksi dengan informasi (Nurhakiki & Yahfizham, 2024).

2.2.3 *You Only Look Once (YOLO)*

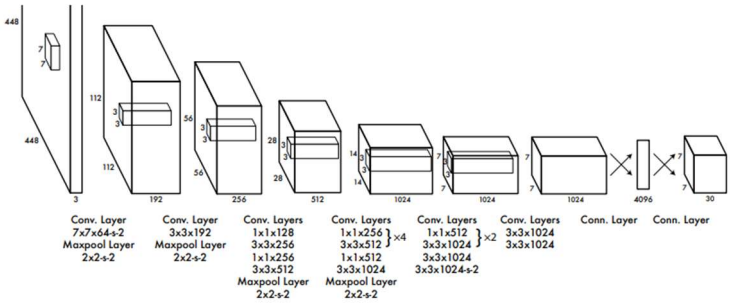
You Only Look Once (YOLO) adalah metode deteksi objek yang diperkenalkan oleh Joseph Redmon dan rekan-rekannya pada tahun 2016,

yang menyelesaikan masalah deteksi objek dengan cara memproses gambar secara *end-to-end* dalam satu jaringan saraf konvolusional (CNN). YOLO membagi gambar menjadi *grid* dan setiap sel *grid* bertanggung jawab untuk mendeteksi objek di area tersebut, memprediksi kelas, koordinat *bounding box*, serta *confidence score* dalam satu langkah.



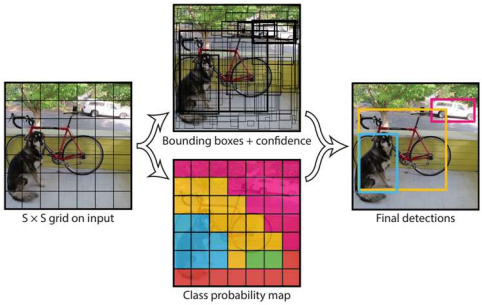
Gambar 2.1 Sistem Deteksi YOLO
Sumber : (Redmon dkk., 2016)

Gambar 2.1 menggambarkan prinsip kerja sistem deteksi objek menggunakan metode YOLO. Untuk awalan, gambar akan diubah ukurannya ke ukuran yang telah ditentukan. Selanjutnya menjalankan CNN untuk membagi gambar menjadi *grid*. Setelah semua prediksi *bounding box* dan skor *confidence score* dihitung, algoritma *Non-Max Suppression* diterapkan untuk menghilangkan redundansi atau tumpang tindih (Redmon dkk., 2016).



Gambar 2.2 Arsitektur YOLO
Sumber : (Redmon dkk., 2016)

Gambar 2.2 menunjukkan arsitektur YOLO, yang menerima input gambar berukuran 448x448x3 dan melewati beberapa lapisan konvolusi dan *pooling* untuk mengekstraksi fitur. Lapisan awal adalah konvolusi 7x7 dengan *stride* 2, diikuti *maxpooling* 2x2 untuk mereduksi dimensi menjadi 112x112x64. Selanjutnya, serangkaian lapisan konvolusi 3x3 menurunkan ukuran fitur (112x112 hingga 28x28) sambil meningkatkan saluran (192 hingga 512). Di bagian akhir, lapisan konvolusi 1x1 dan 3x3 menghasilkan fitur 7x7x1024, yang kemudian diratakan dan diproses melalui *fully connected layer* untuk prediksi kotak pembatas, skor kepercayaan, dan klasifikasi objek. Desain ini memungkinkan deteksi objek *real-time* dengan akurasi tinggi.



Gambar 2.3 Model YOLO
Sumber : Redmon dkk. (2016)

Gambar 2.3 menjelaskan alur kerja sistem deteksi objek pada model YOLO. Untuk awalan, gambar akan dibagi menjadi *grid* S x S. Untuk setiap sel *grid*, model menghasilkan prediksi berupa *bounding box, confidence score, dan class probabilities*. Warna pada peta menunjukkan kemungkinan keberadaan objek tertentu di area tersebut. Terakhir, model memfilter *bounding box* menggunakan algoritma *Non-Max Suppression* untuk menghilangkan kotak yang tumpang tindih dengan skor lebih rendah.

Keunggulan utama YOLO terletak pada kecepatannya, yang memungkinkan deteksi objek secara *real-time*, serta kemampuannya untuk mendeteksi hubungan konteks antara objek yang berbeda dalam gambar. Namun, YOLO juga memiliki kelemahan, seperti kesulitan dalam mendeteksi objek kecil dan akurasi yang menurun pada objek yang saling tumpang tindih. Meskipun demikian, YOLO terus berkembang dengan versi-versi terbaru yang meningkatkan akurasi dan kemampuannya dalam deteksi objek berbagai ukuran. Metode ini telah menjadi salah satu pilihan populer dalam berbagai aplikasi yang memerlukan deteksi objek cepat dan akurat. (Redmon dkk., 2016).

2.2.4 Roboflow

Roboflow adalah platform yang menyediakan alat untuk mempermudah pembuatan dan penerapan model visi komputer. Dengan menggunakan Roboflow, pengguna dapat mengelola *dataset*, melakukan anotasi, dan melatih model visi dengan cepat, bahkan dalam waktu kurang dari 24 jam. Roboflow mendukung berbagai tugas seperti deteksi objek, klasifikasi, segmentasi, dan visi bahasa. Platform ini juga menyediakan fitur augmentasi data, memungkinkan pembuatan versi data yang diperluas untuk meningkatkan keberagaman model. Selain itu, Roboflow menawarkan solusi penyebaran model baik secara *cloud*

maupun *edge*, mendukung perangkat seperti Raspberry Pi dan NVIDIA Jetson. Keamanan dan privasi juga dijaga dengan standar SOC 2 Type 2, serta memungkinkan pengguna untuk memonitor kinerja model yang telah dideploy melalui *dashboard* yang mendetail (Dwyer dkk., 2024).

2.2.5 Google Colab

Google Colab adalah platform berbasis *cloud* yang memungkinkan pengguna untuk membuat dan menjalankan *notebook Jupyter* secara gratis tanpa memerlukan pengaturan perangkat keras. Platform ini menawarkan kemudahan dalam menulis dan menjalankan kode Python untuk keperluan *machine learning*, analisis data, serta pendidikan. Salah satu fitur utama Colab adalah akses gratis ke sumber daya komputasi seperti GPU dan TPU, yang sangat membantu dalam menjalankan model-model *machine learning* yang membutuhkan daya komputasi besar. Selain itu, Google Colab mempermudah kolaborasi dengan memungkinkan pengguna untuk berbagi *notebook* dengan orang lain, yang dapat mengedit atau memberikan komentar secara *real-time*. Fitur lainnya termasuk integrasi dengan Google Drive untuk penyimpanan file dan kemampuan untuk menjalankan kode secara interaktif dalam lingkungan yang mudah digunakan dan dapat diakses dari mana saja (Google, 2024).

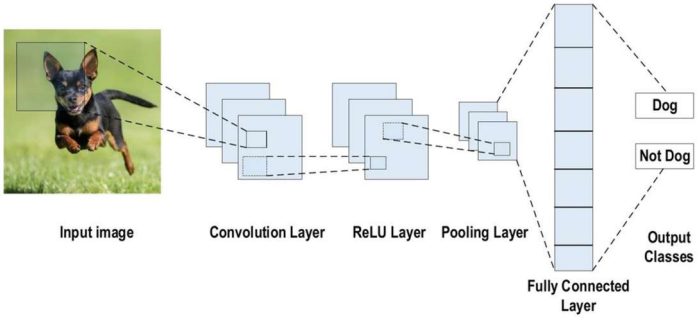
2.2.6 TensorFlow

TensorFlow adalah *framework open-source* yang dikembangkan oleh Google untuk pengembangan dan implementasi model *machine learning*. TensorFlow mendukung berbagai jenis model, termasuk *deep learning* dan *machine learning* tradisional, serta dapat digunakan dalam aplikasi seperti pengenalan gambar dan pemrosesan bahasa alami. Keunggulannya termasuk kemampuannya untuk dijalankan pada berbagai sistem, dari perangkat keras lokal hingga *cloud*, serta dukungan untuk CPU, GPU, dan TPU. TensorFlow juga dilengkapi dengan berbagai alat tambahan, seperti TensorBoard untuk visualisasi, TensorFlow Lite untuk perangkat *mobile*, dan TensorFlow Extended (TFX) untuk pengembangan model skala besar. *Framework* ini telah menjadi pilihan utama di komunitas *machine learning* dan data *science* karena fleksibilitas dan kemampuannya yang luas (Peling dkk., 2024).

2.2.7 *Convolutional Neural Network (CNN)*

Convolutional Neural Networks (CNN) adalah jenis jaringan saraf tiruan yang dirancang khusus untuk memproses data dalam bentuk

grid, seperti gambar. CNN memiliki peran penting dalam *deep learning* dan banyak digunakan dalam tugas-tugas seperti klasifikasi gambar, deteksi objek, dan segmentasi (Alzubaidi dkk., 2021).



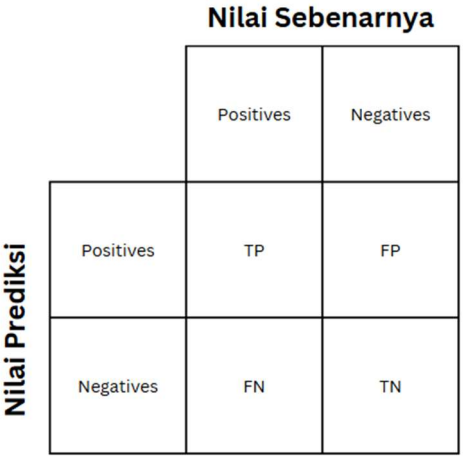
Gambar 2.4 Arsitektur CNN untuk Klasifikasi Gambar
Sumber : (Alzubaidi dkk., 2021)

Terdapat 4 *layer* pada arsitektur CNN untuk klasifikasi gambar, yaitu:

1) *Convolution Layer*
   Layer ini berfungsi untuk mengekstraksi fitur-fitur dari input data. Operasi konvolusi menggunakan filter (kernel) yang bergerak melintasi *input* untuk mendeteksi pola seperti garis, tepi, tekstur, atau pola yang lebih kompleks.
2) *ReLU Layer*
   Layer ini menambahkan *non-linearitas* ke dalam jaringan. Ini penting karena tanpa *non-linearitas*, jaringan hanya bisa memodelkan relasi linier, yang tidak cukup untuk mempelajari pola kompleks.
3) *Pooling Layer*
   Layer ini berfungsi untuk mengurangi dimensi dari *feature map* sambil tetap mempertahankan informasi penting. Ini membantu mengurangi beban komputasi, meningkatkan efisiensi, dan menghindari *overfitting*.
4) *Fully Connected Layer*
   Layer ini menghubungkan setiap neuron di lapisan sebelumnya dengan setiap neuron di lapisan berikutnya. *Fully Connected Layer* bertanggung jawab untuk menghasilkan output akhir dari jaringan CNN, seperti klasifikasi objek.

2.2.8 *Confusion Matrix*

Confusion matrix adalah alat evaluasi yang digunakan untuk mengukur kinerja model klasifikasi dengan membandingkan hasil prediksi model terhadap data aktual. Matriks ini menyediakan gambaran yang jelas mengenai bagaimana model membuat prediksi, dengan membagi hasilnya ke dalam empat kategori: *True Positives (TP), False Positives (FP), True Negatives (TN), dan False Negatives (FN)*.

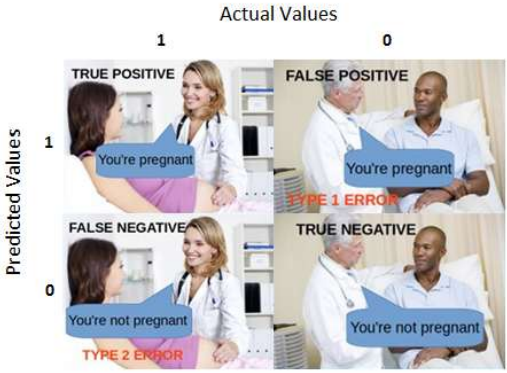


Gambar 2.5 *Confusion Matrix*

1) *True Positives (TP)*: Jumlah contoh yang benar-benar positif dan diprediksi sebagai positif oleh model.
2) *False Positives (FP)*: Jumlah contoh yang sebenarnya negatif tetapi diprediksi sebagai positif.
3) *True Negatives (TN)*: Jumlah contoh yang benar-benar negatif dan diprediksi sebagai negatif oleh model.
4) *False Negatives (FN)*: Jumlah contoh yang sebenarnya positif tetapi diprediksi sebagai negatif.

Dari nilai-nilai ini, berbagai metrik seperti *accuracy, precision, recall, dan F1-score* dapat dihitung. *Confusion matrix* memberikan cara yang komprehensif untuk menganalisis kinerja model, karena tidak hanya mengungkapkan seberapa sering model benar, tetapi juga memberikan wawasan tentang jenis kesalahan yang dibuat model. Hal ini sangat berguna untuk memahami bagaimana model merespons setiap kelas, terutama dalam kasus ketidakseimbangan kelas, di mana model mungkin lebih sering memprediksi kelas mayoritas. *Confusion matrix* juga memungkinkan visualisasi yang lebih mudah, yang membantu dalam interpretasi hasil model, terutama ketika menggunakan grafik *heatmap*

untuk menampilkan distribusi kesalahan secara lebih jelas (Scikit-learn, 2024a).



Gambar 2.6 Contoh *Confusion Matrix*
Sumber : (Narkhede, 2018)

Gambar 2.6 menjelaskan keempat kategori pada *confusion matrix* secara sederhana:

1) *True Positives* (TP): Prediksi model menyatakan bahwa seorang wanita hamil, dan kenyataannya wanita tersebut memang hamil.
2) *False Positives* (FP): Prediksi model menyatakan bahwa seorang pria hamil, padahal secara bio*log*is pria tidak mungkin hamil.
3) *True Neg*atives (TN): Prediksi model menyatakan bahwa seorang pria tidak hamil, dan kenyataannya pria tersebut memang tidak hamil.

False Negatives (FN): Prediksi model menyatakan bahwa seorang wanita tidak hamil, padahal kenyataannya wanita tersebut sedang hamil.

2.2.9 *Mean Average Precision (mAP)*

Mean Average Precision (mAP) adalah metrik yang digunakan untuk mengevaluasi kinerja model deteksi objek.

$$mAP = 1/n \sum_{k=1}^{k=n} AP_k$$

n = jumlah kelas
AP_k = rata-rata presisi dari kelas ke-k

mAP dihitung dengan menghitung *precision* rata-rata untuk setiap kelas pada berbagai ambang batas *recall*, lalu merata-ratakan nilai *precision* tersebut di seluruh kelas. Metrik ini memberikan gambaran seimbang mengenai kemampuan model dalam mendeteksi objek dengan memperhitungkan kesalahan dalam prediksi positif dan negatif. mAP sering digunakan dalam kompetisi dan tugas deteksi objek seperti COCO dan *ImageNet* (Ahmed, 2024).

2.2.10 *Inference Time*

Dalam sistem berbasis kecerdasan buatan, khususnya pada model *deep learning*, istilah inferensi merujuk pada proses ketika model yang telah dilatih sebelumnya digunakan untuk memprediksi atau menganalisis data baru yang masuk. Menurut Goodfellow dkk. (2016), *inference* adalah tahap penerapan model pada data baru untuk menghasilkan *output* prediksi, berbeda dengan tahap pelatihan yang digunakan untuk menyesuaikan parameter model.

Inference time adalah waktu yang dibutuhkan oleh model *deep learning* untuk memproses *input* dan menghasilkan *output* prediksi. Dalam sistem deteksi *real-time* seperti pada proyek ini, *inference time* menjadi parameter penting karena berdampak langsung pada kecepatan sistem dalam merespons kondisi di lapangan. Menurut Redmon dkk. (2016), performa sistem deteksi objek berbasis YOLO sangat dipengaruhi oleh *inference time*, karena tujuan utama dari sistem ini adalah menghasilkan prediksi dengan kecepatan tinggi namun tetap akurat.

Selain itu, diperkenalkan pula konsep *tail quality* oleh Yang dkk. (2024), yang mengukur kualitas hasil deteksi yang masih dapat dipertahankan oleh model dalam batas waktu inferensi tertentu. Konsep ini memberikan gambaran seberapa efektif model dalam menjaga akurasi ketika dituntut bekerja dalam waktu yang sangat terbatas, sebagaimana umum pada sistem deteksi waktu nyata.

Jika sistem memiliki *inference time* sebesar 384,56 ms, maka tiap frame dari kamera CCTV akan diproses dalam waktu kurang dari 0,4 detik. Dengan demikian, sistem dapat mengirim notifikasi pelanggaran hampir seketika setelah pelanggaran terdeteksi.