BAB 2
LANDASAN TEORI

2.1 Bahasa Isyarat Indonesia (BISINDO)

Bahasa Isyarat Indonesia (BISINDO) merupakan bahasa visual yang digunakan oleh komunitas Tuli di Indonesia sebagai sarana komunikasi utama. Bahasa ini berkembang secara alami dalam komunitas pengguna bahasa isyarat dan memiliki struktur linguistik yang memanfaatkan gerakan tangan, ekspresi wajah, serta posisi tubuh untuk menyampaikan makna sehingga lebih natural dalam praktik komunikasi sehari-hari [14]. Oleh karena itu, BISINDO tidak hanya berfungsi sebagai alat komunikasi, tetapi juga menjadi bagian dari identitas budaya komunitas Tuli di Indonesia [15].

2.2 Gesture Recognition

Gesture recognition merupakan teknik dalam *computer vision* yang digunakan untuk mengenali dan menginterpretasikan gerakan tubuh manusia menjadi informasi yang dapat dipahami oleh sistem komputer. Teknologi ini banyak dimanfaatkan pada bidang *human-computer interaction*, sistem kontrol berbasis gerakan, serta penerjemah bahasa isyarat [16]. Dalam sistem pengenalan gesture, proses yang dilakukan umumnya meliputi deteksi tangan, ekstraksi fitur yang merepresentasikan bentuk atau pergerakan tangan, serta proses klasifikasi untuk menentukan jenis gesture yang dikenali [17].

2.3 YOLOv8

YOLOv8 (*You Only Look Once version 8*) merupakan algoritma deteksi objek berbasis *deep learning* yang dikembangkan oleh Ultralytics untuk mendeteksi objek pada citra maupun video secara *real-time*. Model ini menggunakan pendekatan *single-stage detector* sehingga proses klasifikasi objek dan penentuan lokasi *bounding box* dapat dilakukan dalam satu tahap prediksi [1]. Dibandingkan dengan versi sebelumnya, YOLOv8 dirancang dengan arsitektur yang lebih efisien dan performa deteksi yang lebih baik sehingga banyak digunakan pada berbagai aplikasi *computer vision* berbasis video [18, 19].

Selain memiliki kemampuan deteksi yang akurat, YOLOv8 dirancang untuk memenuhi kebutuhan aplikasi *real-time* melalui pendekatan *single-stage detector* yang memungkinkan proses lokalisasi objek dan klasifikasi dilakukan dalam satu kali inferensi. Dibandingkan pendekatan *two-stage detector*, arsitektur ini menghasilkan latensi yang lebih rendah sehingga sesuai digunakan pada aplikasi yang membutuhkan respons cepat, seperti pengenalan gesture bahasa isyarat berbasis kamera. Selain itu, YOLOv8 telah mengadopsi mekanisme *anchor-free detection* dan *decoupled head* yang mampu meningkatkan efisiensi pelatihan sekaligus akurasi deteksi objek pada berbagai skala [1, 20, 21].

2.3.1 Arsitektur YOLOv8

Arsitektur YOLOv8 terdiri dari tiga komponen utama yang bekerja secara berurutan: *Backbone*, *Neck*, dan *Head*. Setiap komponen memiliki peran yang berbeda dalam proses deteksi objek [22, 23].

A Backbone

Backbone berfungsi sebagai ekstraktor fitur utama yang mengubah citra masukan menjadi representasi fitur berhirarki dengan berbagai skala resolusi. YOLOv8 menggunakan arsitektur CSPDarknet yang diperbarui dengan modul C2f (*Cross Stage Partial with 2 Bottleneck*). Modul C2f merupakan pengembangan dari modul C3 pada YOLOv5, yang bekerja dengan membagi *feature map* menjadi dua cabang: satu cabang diproses melalui beberapa blok *bottleneck* secara berurutan, sedangkan cabang lainnya diteruskan langsung, kemudian keduanya digabungkan kembali. Desain ini memungkinkan aliran gradien yang lebih kaya selama proses pelatihan sekaligus meningkatkan efisiensi representasi fitur tanpa menambah beban komputasi secara signifikan [22]. Selain itu, modul SPPF (*Spatial Pyramid Pooling – Fast*) pada lapisan akhir *backbone* memperkaya konteks informasi global dengan menggabungkan hasil *pooling* dari berbagai ukuran *receptive field*.

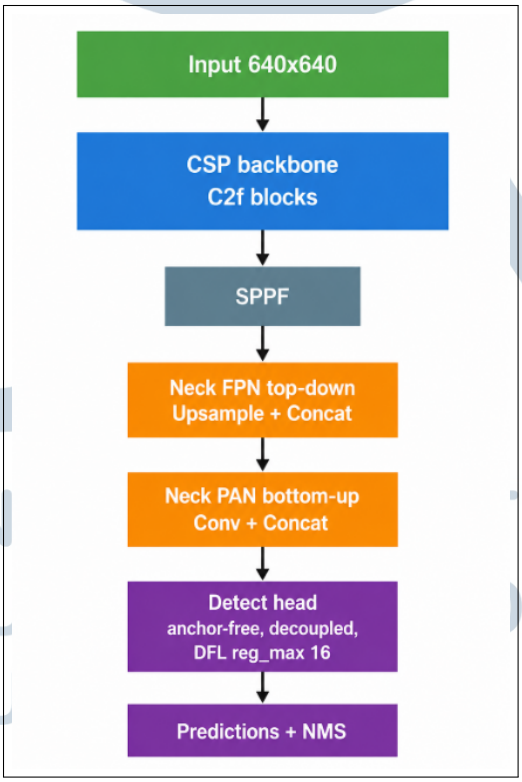
B Neck

Neck menggunakan struktur PANet (*Path Aggregation Network*) yang menggabungkan fitur dari berbagai skala melalui jalur *top-down* dan *bottom-up* secara bersamaan. Jalur *top-down* meneruskan fitur semantik tingkat tinggi dari lapisan dalam ke lapisan dangkal melalui operasi *upsample* dan *concatenate*,

sedangkan jalur *bottom-up* meneruskan informasi lokasi yang presisi dari lapisan dangkal ke lapisan dalam. Kombinasi kedua jalur ini memungkinkan model menggabungkan informasi semantik tingkat tinggi dengan informasi lokasi yang akurat, sehingga mampu mendeteksi objek pada berbagai ukuran secara lebih baik [23].

C Head

Head pada YOLOv8 menggunakan pendekatan *anchor-free* dengan *decoupled head*. Pendekatan *anchor-free* berarti model memprediksi koordinat pusat *bounding box* secara langsung tanpa bergantung pada ukuran *anchor* yang harus ditentukan secara manual sebelum pelatihan, sebagaimana yang diperlukan pada versi YOLO sebelumnya. Sementara itu, *decoupled head* memisahkan cabang prediksi kelas dan cabang prediksi lokasi *bounding box* ke dalam dua jalur konvolusi yang independen, sehingga setiap cabang dapat dioptimalkan secara terpisah. Kombinasi kedua pendekatan ini meningkatkan akurasi dan mempercepat konvergensi selama pelatihan [19].



Gambar 2.1. Arsitektur YOLOv8 yang terdiri atas *Backbone*, *Neck*, dan *Detection Head* (diadaptasi dari [1]).

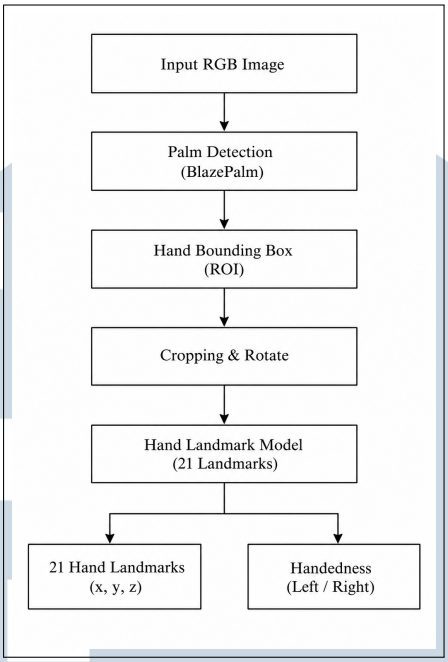
Gambar 2.1 memperlihatkan alur pemrosesan pada arsitektur YOLOv8 yang dimulai dari proses ekstraksi fitur oleh *Backbone*, dilanjutkan dengan agregasi fitur multiskala pada bagian *Neck*, kemudian diakhiri oleh *Detection Head* untuk memprediksi kelas objek beserta koordinat *bounding box*.

Pada penelitian ini, citra masukan yang diproses oleh *Backbone* bukan hanya memuat tekstur tangan dari citra asli, tetapi juga visualisasi 21 titik *hand landmark* hasil proses *keypoint annotation preprocessing*. Selanjutnya, *Neck* menggabungkan fitur-fitur tersebut pada berbagai tingkat resolusi sehingga model dapat mempelajari karakteristik gesture secara lebih komprehensif. Pada tahap akhir, *Detection Head* menghasilkan prediksi berupa kelas gesture BISINDO sekaligus lokasi tangan pada citra. Dengan demikian, setiap komponen dalam arsitektur YOLOv8 memiliki peran yang saling melengkapi sehingga sistem mampu melakukan deteksi gesture secara *real-time* tanpa memerlukan perubahan pada arsitektur dasar YOLOv8.

2.4 MediaPipe

MediaPipe merupakan *framework* yang dikembangkan oleh Google untuk memproses data visual berbasis *machine learning*. Salah satu modulnya, MediaPipe Hands, mampu mendeteksi tangan dan mengekstraksi 21 titik *hand landmark* dari citra RGB secara *real-time*. Titik landmark tersebut merepresentasikan posisi sendi pada jari dan telapak tangan sehingga dapat digunakan untuk menggambarkan pose tangan secara terstruktur [2].

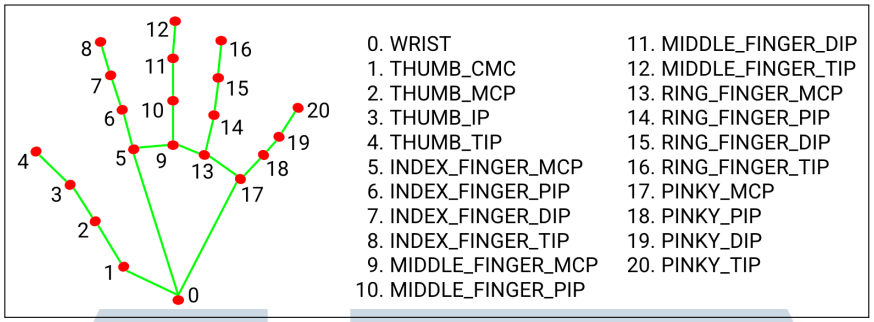
MediaPipe Hands bekerja melalui dua tahap: pertama, deteksi tangan menggunakan model *palm detection* berbasis *BlazePalm* yang menghasilkan *region of interest* (ROI) tangan; kedua, estimasi *landmark* menggunakan model regresi yang memprediksi koordinat 21 titik secara langsung di dalam ROI tersebut [2]. Pipeline dua tahap ini memungkinkan MediaPipe beroperasi secara efisien bahkan pada perangkat dengan sumber daya komputasi terbatas [24].



Gambar 2.2. Arsitektur MediaPipe Hands yang terdiri atas tahap *Palm Detection* dan *Hand Landmark Model* (diadaptasi dari [2]).

Gambar 2.2 menunjukkan bahwa MediaPipe Hands menggunakan arsitektur dua tahap (*two-stage pipeline*). Tahap pertama dilakukan oleh model *Palm Detection* berbasis BlazePalm untuk mendeteksi lokasi telapak tangan dan menghasilkan *bounding box* sebagai *region of interest* (ROI). Selanjutnya, ROI tersebut dipotong (*cropping*) dan disesuaikan orientasinya sebelum diteruskan ke *Hand Landmark Model*. Model ini melakukan regresi terhadap koordinat tiga dimensi (x, y, z) dari 21 titik *hand landmark* serta menentukan orientasi tangan (*handedness*), yaitu tangan kiri atau tangan kanan. Pemisahan proses deteksi tangan dan estimasi landmark membuat MediaPipe Hands mampu bekerja secara efisien dengan latensi rendah sehingga sesuai digunakan pada aplikasi *real-time*.

Pada pengenalan gesture tangan, informasi posisi setiap sendi jari memiliki peranan penting karena mampu merepresentasikan struktur geometris tangan yang relatif stabil terhadap perubahan warna kulit, tekstur, maupun sebagian variasi latar belakang. Oleh karena itu, representasi berbasis *hand landmark* banyak digunakan dalam penelitian pengenalan bahasa isyarat karena lebih menonjolkan informasi pose tangan dibandingkan karakteristik visual yang tidak relevan [17, 25, 26].



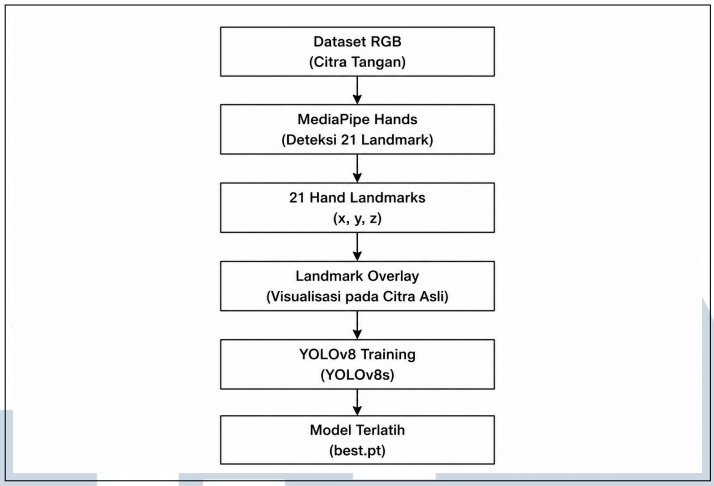
Gambar 2.3. Dua puluh satu titik *hand landmark* pada MediaPipe Hands [2].

Gambar 2.3 menunjukkan 21 titik *hand landmark* yang dihasilkan oleh MediaPipe Hands. Setiap titik merepresentasikan posisi anatomis tertentu pada pergelangan tangan, telapak tangan, serta ruas-ruas jari sehingga mampu menggambarkan struktur geometris tangan secara rinci. Representasi ini banyak dimanfaatkan pada penelitian pengenalan gesture karena menyediakan informasi spasial yang lebih informatif dibandingkan hanya menggunakan citra RGB, terutama untuk membedakan gesture yang memiliki kemiripan bentuk jari [2, 17].

2.4.1 Pipeline YOLOv8 dan MediaPipe

Pada penelitian ini, MediaPipe Hands dan YOLOv8 tidak digunakan secara bersamaan pada tahap ekstraksi fitur, melainkan diintegrasikan melalui tahap *preprocessing*. Citra RGB pada dataset terlebih dahulu diproses menggunakan MediaPipe Hands untuk mendeteksi 21 titik *hand landmark*. Selanjutnya, titik-titik tersebut divisualisasikan kembali sebagai *overlay* pada citra asli sehingga menghasilkan representasi visual yang memuat tekstur tangan sekaligus informasi geometris.

Dataset hasil *overlay* kemudian digunakan sebagai data masukan pada proses pelatihan YOLOv8s. Dengan demikian, YOLOv8 mempelajari karakteristik gesture berdasarkan citra yang telah diperkaya oleh visualisasi landmark tanpa memerlukan perubahan terhadap arsitektur dasarnya. Proses ini menghasilkan model terlatih (*best.pt*) yang selanjutnya digunakan pada implementasi sistem pengenalan gesture secara *real-time*.



Gambar 2.4. Pipeline integrasi MediaPipe Hands dan YOLOv8 pada proses pelatihan model.

Gambar 2.4 menunjukkan alur integrasi MediaPipe Hands dan YOLOv8 pada penelitian ini. Tahap pertama dimulai dengan citra RGB dari dataset yang diproses menggunakan MediaPipe Hands untuk mendeteksi 21 titik *hand landmark*. Landmark yang dihasilkan kemudian divisualisasikan kembali sebagai *overlay* pada citra asli sehingga membentuk representasi visual yang mengandung informasi tekstur tangan sekaligus struktur geometrisnya.

Dataset hasil *overlay* selanjutnya digunakan sebagai masukan pada proses pelatihan YOLOv8s. Pada tahap ini, YOLOv8 mempelajari pola gesture berdasarkan citra yang telah diperkaya oleh visualisasi *landmark* tanpa memerlukan perubahan terhadap arsitektur dasar model. Proses pelatihan menghasilkan model terbaik (*best.pt*) yang kemudian digunakan pada implementasi sistem pengenalan gesture BISINDO secara *real-time*.

2.5 Transfer Learning

Transfer learning merupakan teknik dalam *deep learning* di mana bobot (*weight*) yang telah dipelajari oleh model pada suatu tugas atau dataset sumber digunakan kembali sebagai inisialisasi untuk melatih model pada tugas atau dataset target yang berbeda [15]. Teknik ini sangat berguna ketika dataset target memiliki jumlah data yang terbatas, karena fitur-fitur umum yang telah dipelajari pada dataset sumber seperti tepi, tekstur, dan bentuk dapat dimanfaatkan kembali tanpa perlu dilatih dari awal, sehingga mempercepat proses konvergensi dan mengurangi risiko *overfitting* [20].

Pada penelitian ini, pendekatan *transfer learning* diterapkan dengan memanfaatkan model *pretrained* YOLOv8s (`yolov8s.pt`) yang sebelumnya telah dilatih menggunakan dataset COCO. Model tersebut telah memiliki kemampuan untuk mengekstraksi berbagai fitur visual umum, seperti tepi, tekstur, bentuk, dan pola objek. Oleh karena itu, proses pelatihan tidak dimulai dari bobot awal secara acak (*random initialization*), melainkan menggunakan bobot hasil *pretraining* sebagai titik awal pelatihan.

Sebelum proses pelatihan dilakukan, lapisan *Detection Head* secara otomatis disesuaikan oleh framework Ultralytics agar jumlah keluaran model sesuai dengan jumlah kelas pada dataset penelitian, yaitu 33 kelas gesture BISINDO yang didefinisikan pada berkas `dataset.yaml`. Sementara itu, struktur *Backbone* dan *Neck* tetap dipertahankan untuk memanfaatkan representasi fitur yang telah dipelajari selama proses *pretraining*.

Selanjutnya dilakukan proses *fine-tuning*, yaitu pelatihan ulang menggunakan dataset BISINDO hasil *keypoint annotation preprocessing*. Pada tahap ini seluruh parameter model diperbarui melalui proses *backpropagation* sehingga bobot yang sebelumnya dipelajari dari dataset COCO secara bertahap menyesuaikan karakteristik gesture bahasa isyarat Indonesia. Hasil akhir dari proses ini berupa model terlatih (*best.pt*) yang digunakan pada tahap implementasi sistem pengenalan gesture secara *real-time*.

2.6 Feature Extraction

Feature extraction merupakan proses memperoleh representasi informasi penting dari data mentah sehingga pola yang relevan dapat dipelajari oleh model *deep learning*. Pada pengenalan gesture, fitur yang umum digunakan meliputi bentuk tangan, posisi jari, hubungan spasial antar sendi, serta karakteristik visual lainnya yang membedakan setiap gesture [27].

Pada penelitian ini, proses *feature extraction* tidak dilakukan dengan mengambil koordinat landmark sebagai masukan model secara langsung, melainkan melalui pendekatan *keypoint annotation preprocessing*. MediaPipe terlebih dahulu mendeteksi 21 titik *hand landmark*, kemudian titik dan koneksi antar landmark divisualisasikan di atas citra asli sehingga menghasilkan citra yang mengandung informasi tekstur tangan sekaligus representasi struktur geometrisnya. Pendekatan tersebut memungkinkan model YOLOv8 mempelajari karakteristik visual dan hubungan spasial antar bagian tangan secara bersamaan, tanpa mengubah

arsitektur dasar model.

2.7 Metrik Evaluasi

Metrik evaluasi digunakan untuk mengukur kinerja model dalam melakukan prediksi terhadap data uji. Dalam penelitian berbasis *machine learning* dan *computer vision*, evaluasi model umumnya dilakukan menggunakan beberapa metrik yang diperoleh dari *confusion matrix*, seperti *precision*, *recall*, dan *F1-score*. Metrik tersebut digunakan untuk mengetahui tingkat ketepatan model dalam melakukan klasifikasi data [28, 22].

Selain itu, pada sistem deteksi objek juga digunakan metrik *mean Average Precision* (mAP) untuk mengevaluasi kemampuan model dalam mendeteksi objek berdasarkan kesesuaian antara *bounding box* prediksi dengan data *ground truth*. Metrik ini umum digunakan dalam evaluasi model deteksi objek berbasis YOLO [22, 23].

2.7.1 Confusion Matrix

Confusion matrix merupakan metode evaluasi yang digunakan untuk membandingkan hasil prediksi model dengan data *ground truth*. Pada sistem deteksi objek, *confusion matrix* digunakan sebagai dasar perhitungan berbagai metrik evaluasi seperti *precision*, *recall*, dan *F1-score* [22, 23].

Dalam evaluasi deteksi objek, *True Positive* (TP) menunjukkan objek yang berhasil terdeteksi dengan kelas yang benar dan memenuhi ambang batas *Intersection over Union* (IoU), *False Positive* (FP) menunjukkan objek yang terdeteksi secara salah, sedangkan *False Negative* (FN) menunjukkan objek yang gagal dideteksi oleh model. Komponen *True Negative* (TN) umumnya tidak digunakan secara eksplisit dalam evaluasi model deteksi objek berbasis YOLO karena jumlah area yang tidak mengandung objek sangat besar dibandingkan area objek yang diamati [1].

2.7.2 Intersection over Union (IoU)

Pada sistem deteksi objek, hasil prediksi tidak hanya dinilai berdasarkan ketepatan klasifikasi, tetapi juga berdasarkan kesesuaian lokasi objek yang diprediksi terhadap objek sebenarnya (*ground truth*). Ukuran kesesuaian tersebut dinyatakan menggunakan *Intersection over Union* (IoU), yaitu rasio antara luas

daerah yang saling bertumpang tindih (*intersection*) dengan luas gabungan (*union*) dari *bounding box* prediksi dan *ground truth*. Nilai IoU berada pada rentang 0 hingga 1, di mana semakin besar nilai IoU menunjukkan bahwa lokasi objek hasil prediksi semakin mendekati lokasi objek sebenarnya [1].

IoU dihitung menggunakan Persamaan 2.1.

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \tag{2.1}$$

dengan:

- *Area of Intersection* adalah luas daerah yang saling bertumpang tindih antara *bounding box* hasil prediksi dengan *ground truth*.
- *Area of Union* adalah luas gabungan kedua *bounding box*.

Pada proses evaluasi, nilai IoU digunakan untuk menentukan apakah suatu prediksi dikategorikan sebagai *True Positive* atau *False Positive*. Sebagai contoh, pada evaluasi mAP@0.5 suatu prediksi dianggap benar apabila memiliki nilai IoU minimal 0,50 terhadap *ground truth*.

2.7.3 Precision

Precision digunakan untuk mengukur tingkat ketepatan model dalam memprediksi kelas positif. Metrik ini menunjukkan seberapa banyak prediksi positif yang benar dibandingkan dengan seluruh prediksi positif yang dihasilkan oleh model [28]. Rumus 2.2 menunjukkan cara perhitungan *Precision*.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

dengan:

- TP (*True Positive*) adalah jumlah objek yang berhasil dideteksi dengan kelas yang benar.
- FP (*False Positive*) adalah jumlah objek yang diprediksi sebagai objek target tetapi sebenarnya bukan objek target.

Nilai *precision* berada pada rentang 0 hingga 1. Semakin tinggi nilai *precision*, semakin sedikit prediksi positif yang salah (*False Positive*) sehingga model memiliki tingkat ketepatan prediksi yang semakin baik.

2.7.4 Recall

Recall digunakan untuk mengukur kemampuan model dalam menemukan seluruh data yang termasuk dalam kelas positif. Metrik ini juga sering disebut sebagai *sensitivity*, yaitu kemampuan model dalam mendeteksi seluruh data yang relevan [28]. Rumus 2.3 menunjukkan cara perhitungan *Recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

dengan:

- TP (*True Positive*) adalah jumlah objek yang berhasil dideteksi dengan benar.
- FN (*False Negative*) adalah jumlah objek yang gagal dideteksi oleh model.

Nilai *recall* berada pada rentang 0 hingga 1. Semakin tinggi nilai *recall*, semakin banyak objek yang berhasil ditemukan oleh model sehingga kemungkinan terjadinya *False Negative* menjadi lebih kecil.

2.7.5 F1-Score

F1-score merupakan metrik evaluasi yang menggabungkan nilai *precision* dan *recall* menggunakan rata-rata harmonik sehingga dapat memberikan keseimbangan antara kedua metrik tersebut [28]. Metrik ini sering digunakan ketika distribusi data tidak seimbang karena mampu memberikan gambaran performa model yang lebih komprehensif. Rumus 2.4 menunjukkan cara perhitungan *F1-score*.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

Nilai *F1-Score* berada pada rentang 0 hingga 1. Semakin tinggi nilai *F1-Score*, semakin baik keseimbangan antara kemampuan model dalam menghasilkan

prediksi yang tepat (*Precision*) dan kemampuan model dalam menemukan seluruh objek yang relevan (*Recall*).

2.7.6 Precision–Recall Curve

Kurva *Precision–Recall* merupakan kurva yang menggambarkan hubungan antara nilai *Precision* dan *Recall* ketika nilai *confidence threshold* diubah secara bertahap [28].

Pada setiap nilai *confidence threshold*, model akan menghasilkan jumlah prediksi yang berbeda sehingga diperoleh pasangan nilai *Precision* dan *Recall* yang berbeda pula. Ketika *confidence threshold* diturunkan, model cenderung menghasilkan lebih banyak prediksi sehingga nilai *Recall* meningkat, namun nilai *Precision* dapat menurun akibat bertambahnya jumlah *False Positive*. Sebaliknya, ketika *confidence threshold* dinaikkan, model menjadi lebih selektif dalam menghasilkan prediksi sehingga nilai *Precision* meningkat, tetapi nilai *Recall* dapat menurun karena sebagian objek tidak lagi terdeteksi.

Kurva *Precision–Recall* digunakan sebagai dasar perhitungan nilai *Average Precision* (AP). Semakin besar luas area di bawah kurva tersebut, semakin baik kemampuan model dalam mempertahankan nilai *Precision* pada berbagai tingkat *Recall*.

2.7.7 Average Precision (AP)

Average Precision (AP) merupakan ukuran performa deteksi objek untuk setiap kelas yang dihitung berdasarkan luas area di bawah kurva *Precision–Recall*. Nilai AP menggambarkan kemampuan model dalam mempertahankan nilai *Precision* pada berbagai tingkat *Recall*. Semakin besar luas area di bawah kurva *Precision–Recall*, semakin tinggi nilai AP yang diperoleh [1].

Secara matematis, Average Precision dihitung menggunakan Persamaan 2.5.

$$AP = \int_0^1 P(R)dR \tag{2.5}$$

dengan:

- $P(R)$ merupakan fungsi *Precision* terhadap *Recall*.
- Nilai *Recall* berada pada rentang 0 hingga 1.

Persamaan tersebut menunjukkan bahwa nilai AP merupakan luas area di bawah kurva *Precision–Recall*. Pada implementasinya, perhitungan AP dilakukan secara numerik menggunakan seluruh pasangan nilai *Precision* dan *Recall* yang dihasilkan model pada berbagai nilai *confidence threshold*.

2.7.8 Mean Average Precision (mAP)

Mean Average Precision (mAP) merupakan rata-rata nilai *Average Precision* (AP) dari seluruh kelas yang dievaluasi. Metrik ini digunakan sebagai ukuran utama performa model deteksi objek karena mempertimbangkan kemampuan klasifikasi sekaligus ketepatan lokalisasi objek [1].

Nilai mAP dihitung menggunakan Persamaan 2.6.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.6}$$

dengan:

- N merupakan jumlah kelas yang dievaluasi.
- AP_i merupakan nilai *Average Precision* pada kelas ke-i.

Pada penelitian deteksi objek berbasis YOLO, nilai mAP umumnya dilaporkan dalam dua bentuk, yaitu mAP@0.5 dan mAP@0.5:0.95.

mAP@0.5 dihitung menggunakan satu nilai ambang *Intersection over Union* (IoU), yaitu sebesar 0,50. Dengan kata lain, suatu prediksi dikategorikan sebagai benar apabila memiliki nilai IoU minimal 0,50 terhadap *ground truth*.

Sementara itu, mAP@0.5:0.95 dihitung menggunakan sepuluh nilai ambang IoU, yaitu 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, dan 0.95 sebagaimana standar evaluasi COCO [1]. Nilai AP dihitung pada setiap ambang IoU tersebut, kemudian dirata-ratakan sehingga diperoleh nilai mAP@0.5:0.95.

Karena menggunakan kriteria IoU yang lebih ketat, nilai mAP@0.5:0.95 umumnya lebih rendah dibandingkan nilai mAP@0.5. Oleh karena itu, mAP@0.5:0.95 dianggap mampu memberikan gambaran performa model secara lebih komprehensif, terutama dalam mengevaluasi ketepatan lokasi *bounding box*.