BAB 2
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu yang berkaitan dengan deteksi konten promosi judi *online* pada media sosial telah dilakukan dengan berbagai pendekatan, mulai dari deteksi objek berbasis *deep learning* hingga sistem multimodal yang menggabungkan deteksi visual, ekstraksi teks, dan klasifikasi semantik. Kajian terhadap penelitian-penelitian tersebut dilakukan untuk mengidentifikasi celah yang menjadi landasan pengembangan sistem pada penelitian ini. Ringkasan penelitian terdahulu disajikan pada Tabel 2.1.

Tabel 2.1. Penelitian Terdahulu

No	Peneliti (Tahun)	Judul	Metode	Hasil Terbaik	Keterbatasan
1	Aldamawan (2023)	Deteksi Iklan Judi Online pada Gambar dengan Algoritma Faster R-CNN	Faster R-CNN ResNet-101	*Average Precision* 90,1%, *Average Recall* 92,1% pada rasio data 90:10 dan *num_steps* 10.000	Hanya mendeteksi elemen visual; tidak menganalisis konten teks pada *watermark*
2	Ritonga (2025)	Implementasi Algoritma *Deep Learning* Faster R-CNN untuk Mendeteksi Kemungkinan Iklan Judi Online Sisipan pada Video Reels Instagram	Faster R-CNN + PyTorch	mAP 0,7849; *Precision* 0,9900; F1-*Score* 0,9011; rata-rata waktu respons ±4,5 detik	Terbatas pada deteksi visual; tidak menganalisis teks *watermark*; hanya memproses 15 detik pertama video
3	Maldini (2025)	Implementasi Faster R-CNN dan OCR dengan Pendekatan *Natural Language Processing* untuk Deteksi Iklan Judi Online	Faster R-CNN (ResNet-50 + FPN) + TrOCR + BERT	mAP@0.50 98,1%; CER TrOCR 4,6%; akurasi BERT 99%	Kebutuhan komputasi tinggi; kurang efisien untuk moderasi konten *real-time*
4	Maldini et al. (2025)	*Multimodal Detection of Covert Online Gambling Advertisements Using Faster R-CNN and Tr-OCR*	Faster R-CNN (ResNet-50 + FPN) + TrOCR + BERT (multimodal)	*Average Precision* 98,1%; CER 4,6%; WER 29%; akurasi BERT 99%	Kebutuhan komputasi tinggi; beban model besar menghambat penerapan *real-time*

Lanjut pada halaman berikutnya...

Tabel 2.1 Penelitian Terdahulu (lanjutan)

No	Peneliti (Tahun)	Judul	Metode	Hasil Terbaik	Keterbatasan
5	Tokang (2025)	Implementasi YOLOv11 dengan Teknik *Quantization* untuk Deteksi Iklan Judi Online di Instagram	YOLOv11n/s + PTQ (FP32/FP16/INT8) pada TensorRT, OpenVINO, TFLite	OpenVINO INT8: F1-*Score* 99,9%, mAP@0.5:0.95 76,7%; TensorRT INT8: inferensi 3,5 ms	Hanya mendeteksi elemen visual; tidak disertai analisis teks *watermark*
6	Faisal et al. (2025)	Implementasi Algoritma YOLO11 untuk Deteksi Logo Situs Judi Online pada Konten Video Reel Instagram	YOLO11 + augmentasi fotometrik	mAP50 72,9%; *Precision* 90,4% untuk kelas logo judi *online*	Hanya mendeteksi logo; tidak menganalisis konten teks pada *watermark*

Berdasarkan Tabel 2.1, terdapat dua pola utama yang dapat diidentifikasi. Pertama, penelitian yang hanya berfokus pada deteksi visual [4–7] mampu mengidentifikasi keberadaan elemen *watermark* atau logo, namun tidak menganalisis konten teks yang terkandung di dalamnya. Kedua, penelitian yang mengintegrasikan deteksi visual dan analisis teks [8, 9] berhasil mencapai akurasi yang sangat tinggi, tetapi mengandalkan model besar seperti TrOCR dan BERT yang membutuhkan sumber daya komputasi tinggi sehingga kurang optimal untuk kebutuhan moderasi konten secara *real-time*.

Perbandingan aspek teknis antara penelitian terdahulu dan penelitian ini disajikan secara lebih ringkas pada Tabel 2.2.

Tabel 2.2. Perbandingan Aspek Teknis Penelitian Terdahulu dengan Penelitian Ini

Aspek	Penelitian Terdahulu						Penelitian Ini
	Aldamawan (2023)	Ritonga (2025)	Maldini (2025)	Maldini et al. (2025)	Tokang (2025)	Faisal et al. (2025)	
Deteksi objek visual (*watermark*/logo)	✓	✓	✓	✓	✓	✓	✓
Konten video Reels Instagram	×	✓	×	×	×	✓	✓
Ekstraksi teks dari *watermark* (OCR)	×	×	✓	✓	×	×	✓
Klasifikasi teks hasil OCR	×	×	✓	✓	×	×	✓
Model deteksi ringan (YOLO)	×	×	×	×	✓	✓	✓
Klasifikasi teks berbasis TF-IDF + SVM	×	×	×	×	×	×	✓

Berdasarkan kedua tabel di atas, dapat disimpulkan bahwa belum terdapat penelitian yang secara bersamaan menggabungkan model deteksi objek ringan berbasis YOLO, ekstraksi teks menggunakan OCR, dan klasifikasi teks berbasis TF-IDF dan SVM dalam satu sistem terintegrasi yang difokuskan pada konten video Reels Instagram berbahasa Indonesia. Celah tersebut menunjukkan bahwa pendekatan hibrida yang menggabungkan ketiga komponen tersebut berpotensi menghasilkan sistem yang lebih seimbang antara akurasi dan efisiensi komputasi, sekaligus mampu menganalisis konten *watermark* secara komprehensif dibandingkan pendekatan yang hanya mengandalkan deteksi visual semata.

2.2 Watermark Judi Online pada Konten Digital

Watermark judi online merupakan elemen visual berupa teks, logo, simbol, atau kombinasi keduanya yang sengaja disisipkan pada konten digital dengan tujuan mempromosikan aktivitas perjudian daring [19]. Berbeda dengan *watermark* konvensional yang umumnya berfungsi sebagai penanda kepemilikan atau bentuk perlindungan hak cipta, *watermark* judi online dirancang khusus sebagai media promosi yang mengarahkan pengguna menuju situs, aplikasi, maupun layanan perjudian tertentu [8]. Praktik semacam ini banyak ditemukan pada konten media sosial, khususnya video pendek yang diunggah melalui platform seperti Instagram Reels, TikTok, Facebook Reels, dan YouTube Shorts.

Penyisipan *watermark* promosi judi online umumnya memanfaatkan tingginya tingkat konsumsi konten video pendek di kalangan masyarakat. Pelaku biasanya menyematkan teks promosi berupa nama situs, nomor kontak, akun media sosial, tautan tertentu, maupun kata-kata persuasif dan nama situs seperti “slot”, “gigacuan”, “maxwin”, “jackpot”, “toto”, “LUCKYBET89”, “HOKIRAJA”, dan “SENSA138”, dan berbagai istilah lain yang lazim digunakan dalam promosi perjudian daring [8]. Elemen-elemen tersebut ditempatkan pada area tertentu dalam video sehingga tetap terlihat oleh penonton tanpa mengganggu isi utama konten. Contoh *watermark* promosi judi online pada konten digital dapat dilihat pada Gambar 2.1.

Gambar 2.1. Contoh *Watermark* Promosi Judi Online pada Konten Digital

Sumber: Dokumentasi pribadi

Fenomena ini semakin marak ditemukan di berbagai platform media sosial. Pelaku umumnya memanfaatkan video hiburan yang sedang populer sebagai media penyisipan informasi promosi perjudian secara terselubung. Penyisipan dilakukan melalui teks, logo, maupun nama situs yang diletakkan pada bagian tertentu dari video. Strategi ini ditempuh untuk memperluas jangkauan promosi sekaligus menghindari deteksi langsung oleh sistem moderasi konten maupun pengguna umum [19, 20]. Lebih jauh, keberadaan *watermark* semacam ini tidak hanya berfungsi sebagai sarana promosi, tetapi juga berpotensi melanggar ketentuan hukum yang mengatur penyebaran konten perjudian di ruang digital [19, 20].

Secara visual, *watermark* judi online memiliki karakteristik yang beragam. Teks *watermark* dapat hadir dalam berbagai ukuran, jenis huruf, warna, serta tingkat transparansi yang berbeda-beda. Penempatannya pun bervariasi, mulai dari sudut video, bagian tengah, hingga tepi layar, bahkan ada yang mengikuti pergerakan objek tertentu dalam video. Pada sejumlah kasus, *watermark* disajikan dalam bentuk animasi bergerak, yang menjadikannya lebih sulit dideteksi dibandingkan teks statis [8].

2.3 Deteksi Objek

Deteksi objek merupakan salah satu cabang utama dalam bidang *computer vision* yang bertujuan untuk mengidentifikasi dan melokalisasi objek pada citra atau video [21]. Teknologi ini banyak digunakan pada berbagai aplikasi, seperti

sistem pengawasan, navigasi robotika, dan sistem bantu bagi tunanetra karena kemampuannya dalam memahami lingkungan visual secara otomatis [21, 22].

Seiring perkembangan *deep learning*, berbagai arsitektur deteksi objek telah dikembangkan dan secara umum dapat dikelompokkan menjadi model berbasis *Convolutional Neural Network* (CNN) dan Transformer [21]. Model berbasis CNN memanfaatkan lapisan konvolusi untuk mengekstraksi fitur visual yang merepresentasikan objek dalam citra [21, 23].

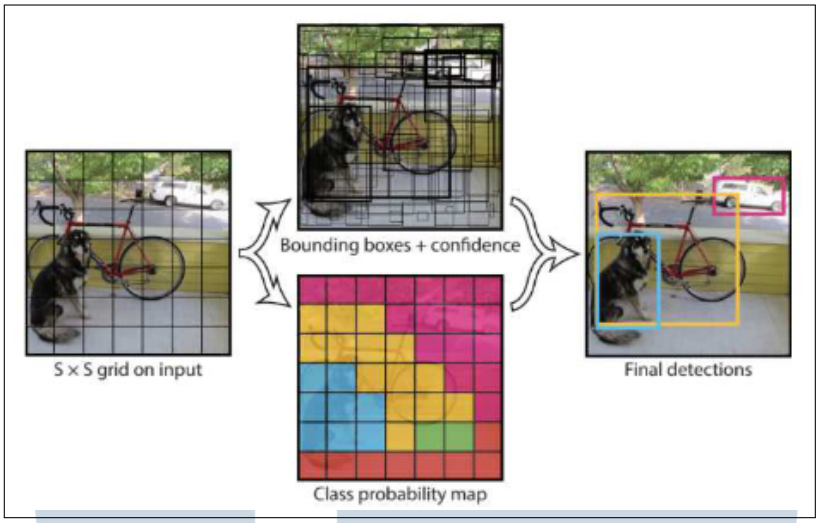
Pada arsitektur CNN, terdapat dua pendekatan utama, yaitu *two-stage detector* dan *one-stage detector*. Pendekatan *two-stage detector*, seperti R-CNN dan Faster R-CNN, menghasilkan *region proposal* terlebih dahulu sebelum melakukan klasifikasi objek sehingga mampu mencapai akurasi yang tinggi, tetapi memiliki waktu inferensi yang relatif lebih lambat [21, 23]. Sebaliknya, *one-stage detector* seperti *You Only Look Once* (YOLO) dan *Single Shot Detector* (SSD) secara langsung memprediksi *bounding box* dan kelas objek dalam satu tahap sehingga lebih cepat dan cocok untuk aplikasi *real-time* [21–23].

Selain CNN, arsitektur Transformer juga mulai digunakan dalam deteksi objek, salah satunya melalui *Detection Transformer* (DETR) yang memanfaatkan mekanisme *self-attention* untuk menangkap hubungan global pada citra dan melakukan deteksi objek secara *end-to-end* [21].

2.4 You Only Look Once (YOLO)

You Only Look Once (YOLO) adalah salah satu algoritma yang digunakan untuk deteksi objek secara *real-time* dan segmentasi gambar. Algoritma ini dikembangkan oleh Joseph Redmond dan Ali Farhadi di University of Washington dan pertama kali dirilis pada tahun 2016 [24]. YOLO memanfaatkan arsitektur *Convolutional Neural Network* (CNN) untuk melakukan deteksi objek pada gambar atau video dalam satu proses inferensi jaringan saraf tunggal yang efisien dan cepat, tanpa melalui tahap *region proposal* seperti pada metode dua tahap [24].

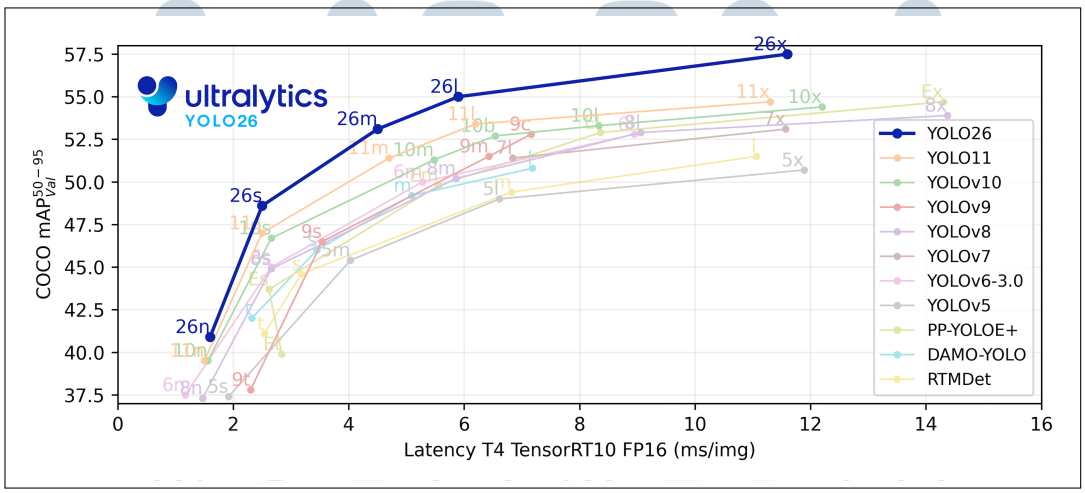
Algoritma YOLO menerapkan satu jaringan saraf pada keseluruhan gambar, membaginya menjadi beberapa wilayah dan memprediksi *bounding box* serta probabilitas untuk setiap wilayah tersebut. Seperti yang ditunjukkan pada Gambar 2.2, gambar masukan dibagi menjadi *grid* berukuran $S \times S$. Untuk setiap sel pada *grid*, model memprediksi B *bounding box*, nilai *confidence* untuk setiap kotak, dan probabilitas kelas C. Prediksi ini dikodekan sebagai tensor berukuran $S \times S \times (B \times 5 + C)$ [24].



Gambar 2.2. Model Deteksi YOLO

Sumber: [24]

Sejak diperkenalkan pada tahun 2016, YOLO telah mengalami berbagai pengembangan yang bertujuan meningkatkan kinerja deteksi objek baik dari sisi akurasi maupun kecepatan inferensi. Berbagai versi YOLO dikembangkan dengan penyempurnaan arsitektur dan teknik optimasi sehingga mampu memberikan performa yang semakin baik pada berbagai tugas deteksi objek. Perkembangan performa antar versi YOLO dapat dilihat pada Gambar 2.3.



Gambar 2.3. Perbandingan Performa Antar Versi YOLO

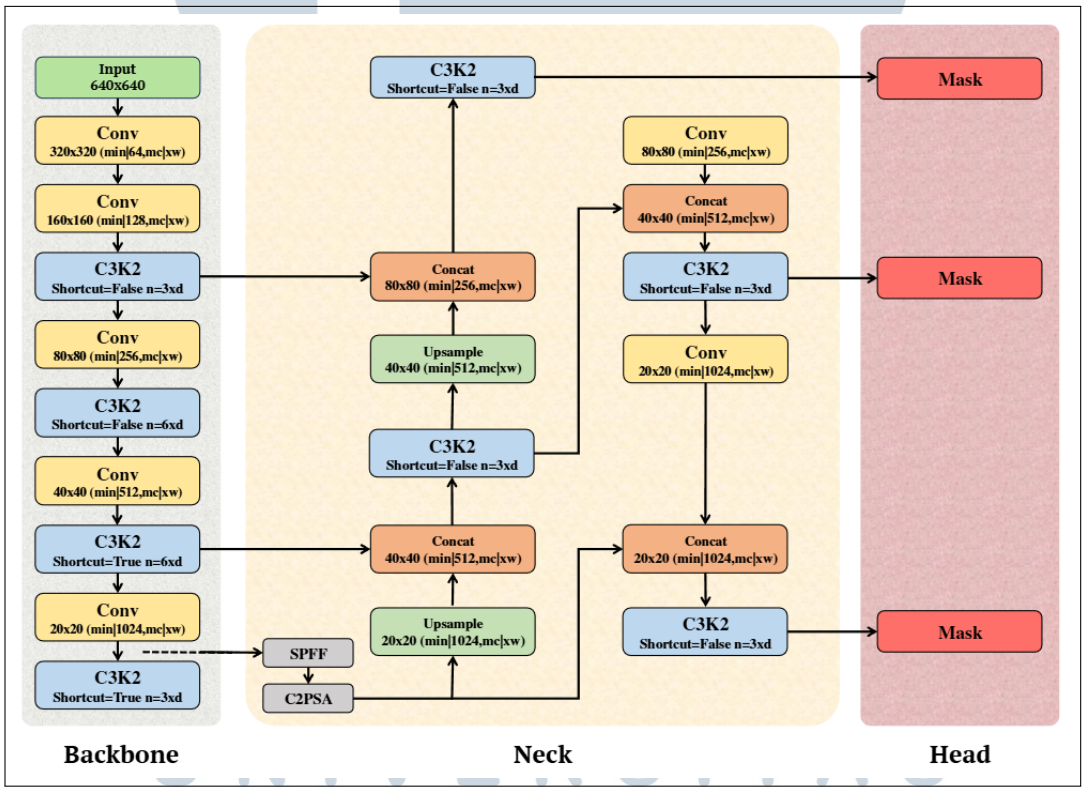
Sumber: [25]

Perkembangan berbagai versi YOLO menunjukkan adanya upaya berkelanjutan untuk meningkatkan kemampuan deteksi objek. Salah satu

pengembangan terbaru dari keluarga YOLO adalah YOLOv11 yang akan dibahas pada subbab berikutnya.

2.4.1 YOLOv11

YOLOv11 merupakan versi terbaru dari keluarga algoritma YOLO yang hadir dengan berbagai pembaruan signifikan, termasuk peningkatan arsitektur *backbone*, mekanisme fusi fitur yang lebih baik, dan strategi pelatihan adaptif [26]. YOLOv11 dirancang untuk memberikan akurasi yang lebih tinggi dan deteksi *multi-scale* yang lebih efektif, khususnya untuk objek-objek yang sulit dikenali [26]. Arsitektur YOLOv11 terdiri dari tiga bagian utama yaitu *backbone*, *neck*, dan *head* [26], sebagaimana ditunjukkan pada Gambar 2.4.



Gambar 2.4. Arsitektur YOLOv11
Sumber: [27]

Inovasi utama dalam arsitektur YOLOv11 terletak pada penggunaan tiga komponen kunci. Pertama, blok *Cross Stage Partial with 2 kernel size* (C3K2) yang menggantikan blok-blok konvolusi yang lebih berat secara komputasi. Blok ini terdiri dari dua konvolusi yang lebih kecil untuk mengurangi biaya komputasi

sambil tetap mempertahankan performa, sehingga model dapat memproses informasi spasial dengan lebih efisien tanpa mengorbankan kecepatan inferensi [26–28]. Kedua, modul *Spatial Pyramid Pooling Fast* (SPPF) yang diwarisi dari YOLOv8, berfungsi untuk mengumpulkan fitur pada berbagai skala sehingga meningkatkan representasi objek pada ukuran yang bervariasi [26–28]. Ketiga, blok *Cross-Stage Partial with Spatial Attention* (C2PSA) yang merupakan penambahan baru dalam YOLOv11. Blok ini mengintegrasikan mekanisme *spatial attention* yang memungkinkan model untuk fokus pada wilayah-wilayah penting dalam gambar yang paling relevan untuk deteksi, sehingga meningkatkan performa pada objek-objek kecil maupun objek yang saling berdekatan [26–28].

Komponen *neck* pada YOLOv11 dirancang untuk menggabungkan *feature map* dari berbagai resolusi dan meneruskannya ke *detection head*, dengan mengintegrasikan blok C3K2 untuk meningkatkan kecepatan dan performa agregasi fitur. Sementara itu, bagian *head* bertanggung jawab menghasilkan prediksi akhir berupa *bounding box*, probabilitas kelas, dan *confidence score* pada tiga skala deteksi, yaitu kecil (P3), sedang (P4), dan besar (P5), yang memungkinkan deteksi objek pada berbagai ukuran [26].

2.5 Augmentasi Data

Data augmentation atau augmentasi data merupakan sebuah strategi yang diterapkan untuk memperbanyak dan memperkaya variasi data latih tanpa perlu menambah jumlah data asli, dengan cara menerapkan berbagai transformasi terhadap citra yang sudah ada [29]. Teknik ini umumnya diterapkan untuk mengatasi permasalahan *overfitting* maupun ketidakseimbangan kelas (*data imbalance*) pada dataset yang berukuran terbatas, sekaligus membantu model mempelajari lebih banyak variasi visual dari objek target sehingga kemampuan generalisasinya terhadap data baru meningkat [29, 30].

Secara umum, teknik augmentasi data pada deteksi objek berbasis YOLO dapat dikelompokkan menjadi dua kategori utama, yaitu transformasi geometris dan transformasi fotometrik [31]. Transformasi geometris mengubah posisi, orientasi, maupun skala objek pada citra, di antaranya meliputi rotasi (*rotation*), pergeseran (*translation*), perubahan skala (*scaling*), pembalikan (*flip*), serta pergeseran sudut (*shear*) [30, 31]. Sementara itu, transformasi fotometrik mengubah karakteristik warna dan intensitas piksel pada citra tanpa mengubah posisi objek, seperti penyesuaian kecerahan (*brightness*), kontras (*contrast*), rona

dan saturasi warna (*hue/saturation* atau HSV), penambahan derau (*Gaussian noise*), maupun pengaburan citra (*Gaussian blur*) [31]. Kedua kategori transformasi ini dipilih dan disesuaikan berdasarkan kondisi visual nyata yang ingin disimulasikan, misalnya rotasi kecil untuk mengakomodasi kemiringan pengambilan gambar, atau penyesuaian kecerahan untuk mensimulasikan kondisi pencahayaan yang berbeda-beda [31].

Selain transformasi tunggal, terdapat pula teknik augmentasi komposit yang menggabungkan lebih dari satu citra latih sekaligus untuk memperkaya representasi fitur antarsampel, di antaranya *mosaic*, *mixup*, dan *cutmix* [30]. Teknik *mosaic* bekerja dengan menggabungkan beberapa citra latih menjadi satu bingkai baru sehingga memperluas variasi skala dan konteks latar belakang objek. Teknik *mixup* mengombinasikan dua citra secara acak melalui rata-rata piksel berbobot untuk membantu mengurangi *overfitting*, sedangkan *cutmix* menempelkan potongan salah satu citra ke citra lain untuk meningkatkan ketahanan model terhadap kondisi objek yang saling tumpang tindih (*occlusion*) [30].

2.6 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) adalah teknologi yang mampu membaca teks dari gambar dan mengubahnya menjadi data digital yang dapat disimpan atau diolah lebih lanjut [32, 33]. Secara teknis, OCR merupakan proses mengubah gambar huruf menjadi karakter *American Standard Code for Information Interchange* (ASCII) yang dapat dikenali oleh komputer [34].

OCR bekerja dengan memanfaatkan teknik pengolahan citra digital (*image processing*) untuk mendeteksi karakter huruf pada gambar, kemudian mengubahnya menjadi teks yang terbaca oleh sistem [32]. Salah satu metode yang umum digunakan dalam OCR adalah *template matching*, yaitu metode yang bekerja dengan cara mendeteksi karakter huruf dari citra yang telah diproses sebelumnya, kemudian mencocokkannya dengan *template* karakter yang tersedia dalam sistem [32]. Tingkat kecocokan ditentukan menggunakan metode *Normalized Cross Correlation* (NCC). Persentase kecocokan dengan nilai tertinggi akan menentukan jenis karakter yang dikenali [34].

Secara umum, proses OCR terdiri dari beberapa tahapan. Pertama, tahap *preprocessing* citra yang mencakup pemilihan *Region of Interest* (ROI), konversi *grayscale*, peningkatan kontras, penghilangan *noise*, serta penyesuaian ukuran gambar [32]. Kedua, pencarian baris dan kata pada citra dengan

menggunakan penyaringan gumpalan dan konstruksi garis. Ketiga, deteksi dan pemotongan kata menjadi karakter menggunakan *pitch*. Keempat, pemotongan dan penyambungan karakter dari gumpalan citra. Terakhir, tahap *template matching* untuk mencocokkan karakter hasil potongan dengan referensi yang tersedia [34]. Kualitas citra masukan sangat mempengaruhi hasil pembacaan OCR. Faktor-faktor seperti tingkat fokus kamera, intensitas pencahayaan, dan orientasi dokumen terhadap kamera berperan penting dalam menentukan akurasi pengenalan karakter [34].

2.7 PaddleOCR

PaddleOCR adalah *toolkit Optical Character Recognition* (OCR) *open-source* yang dikembangkan oleh PaddlePaddle, yaitu *framework deep learning* milik Baidu [35, 36]. Sistem ini dirancang untuk mendeteksi dan mengenali teks dalam berbagai bahasa dengan akurasi tinggi, kecepatan pemrosesan yang baik, serta dukungan terhadap lebih dari 80 bahasa, termasuk bahasa dengan karakter kompleks [36]. PaddleOCR menyediakan model *pre-trained* siap pakai seperti seri PP-OCR yang telah dioptimasi untuk menyeimbangkan kecepatan dan akurasi [36].

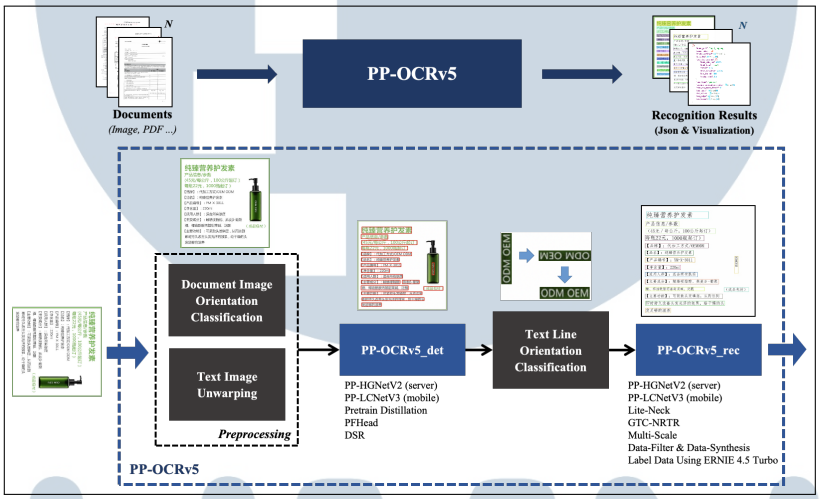
Secara arsitektur, PaddleOCR menggunakan pendekatan *pipeline* dua tahap yang menggabungkan dua komponen utama [37]. Pertama, untuk deteksi teks, PaddleOCR menggunakan arsitektur berbasis CNN seperti *Differentiable Binarization* (DB) dengan *backbone* ResNet atau MobileNet yang dimodifikasi [36]. Kedua, untuk pengenalan teks, PaddleOCR mengandalkan *Convolutional Recurrent Neural Network* (CRNN). Pada arsitektur ini, CNN berfungsi sebagai ekstraktor fitur visual yang kemudian dilanjutkan dengan lapisan BiLSTM untuk pemodelan konteks urutan karakter [36]. Pendekatan ini dilengkapi dengan *decoder Connectionist Temporal Classification* (CTC) untuk mengubah fitur menjadi teks akhir [36]. Versi terbarunya, SVTR, bahkan menggantikan LSTM dengan *transformer* untuk menangani dependensi jarak jauh secara lebih efektif [36].

2.7.1 PP-OCRv5

PP-OCRv5 merupakan salah satu versi dari seri PP-OCR yang diperkenalkan oleh Baidu dan dipresentasikan dalam konferensi CVPR 2026 [37]. PP-OCRv5 adalah sistem OCR dua tahap yang sangat efisien dengan total hanya 5 juta parameter, namun mampu mencapai performa yang kompetitif dengan

model *Vision-Language Model* (VLM) berukuran miliaran parameter pada berbagai *benchmark* OCR standar [37].

Arsitektur PP-OCRv5 mengikuti *pipeline* dua tahap yang telah terbukti efektif, terdiri dari modul deteksi teks dan modul pengenalan teks, sebagaimana ditunjukkan pada Gambar 2.5. Pada modul deteksi teks, PP-OCRv5 menggunakan algoritma *Differentiable Binarization* (DB) untuk lokalisasi teks yang andal, dengan *backbone* PP-LCNetV3 untuk ekstraksi fitur yang efisien [37]. Modul pengenalan teks mengadopsi arsitektur hibrida SVTR LCNet yang menggabungkan kekuatan *framework* SVTR dengan *backbone* PP-LCNetV3, serta menerapkan strategi *Guided Training of CTC* (GTC) untuk pemodelan urutan yang lebih efektif [37].



Gambar 2.5. *Framework* PP-OCRv5

Sumber: [37]

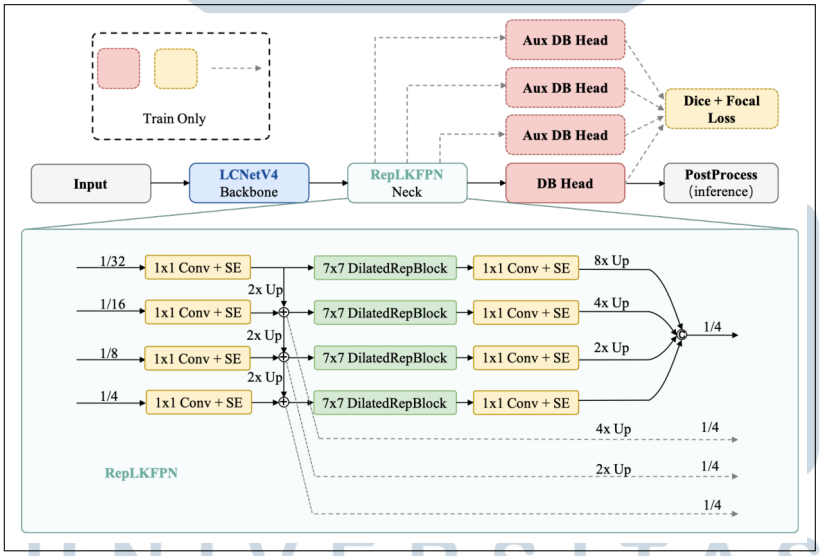
Keunggulan utama PP-OCRv5 terletak pada pendekatan *data-centric* yang sistematis dalam proses pengembangannya. Berbeda dari pendekatan konvensional yang berfokus pada inovasi arsitektur, PP-OCRv5 mengoptimalkan data pelatihan berdasarkan tiga dimensi kunci [37]. Pertama, *data difficulty*, yaitu pengukuran tingkat kesulitan sampel menggunakan skor *confidence* model, di mana sampel dengan rentang skor 0,95–0,97 terbukti memberikan performa terbaik sebagai *"sweet spot"* yang mengandung sampel menantang namun berlabel benar [37]. Kedua, *data accuracy*, yaitu pengukuran dampak *noise* pada label data, di mana model terbukti memiliki ketahanan yang sangat baik bahkan ketika tingkat kesalahan label mencapai 20% [37]. Ketiga, *data diversity*, yaitu keluasan ruang fitur visual yang dicakup oleh data pelatihan, di mana peningkatan keragaman dari 200 menjadi 1.000 klaster menghasilkan peningkatan akurasi sebesar 5,38

poin persentase [37]. Melalui metodologi *data-centric* tersebut, PP-OCRv5 dilatih menggunakan dataset sebesar 22,6 juta sampel yang mencakup berbagai skenario, mulai dari teks cetak, tulisan tangan, hingga teks artistik dalam berbagai bahasa [37].

2.7.2 PP-OCRv6

PP-OCRv6 merupakan versi terbaru dari seri PP-OCR yang diperkenalkan oleh Baidu pada tahun 2026 [38]. PP-OCRv6 adalah sistem OCR ringan yang menggabungkan inovasi arsitektur dengan optimasi *data-centric*, mencakup parameter berkisar dari 1,5 juta hingga 34,5 juta parameter [38].

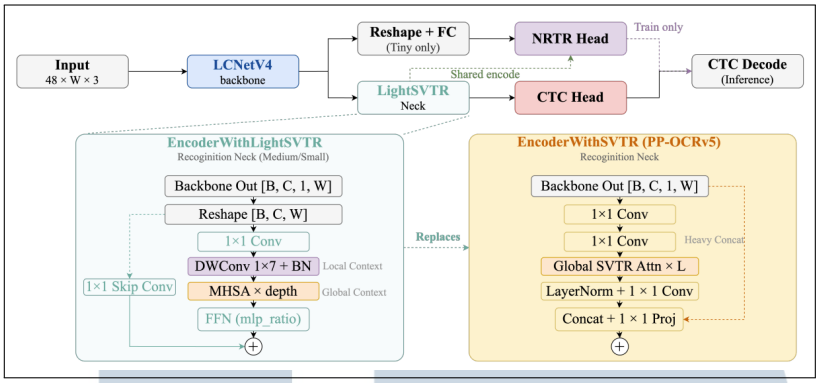
Arsitektur PP-OCRv6 mengikuti *pipeline* dua tahap yang terdiri dari modul deteksi teks dan modul pengenalan teks. Pada modul deteksi teks, sebagaimana ditunjukkan pada Gambar 2.6, PP-OCRv6 menggunakan *backbone* LCNetV4 dengan *neck* RepLKFPN, kepala DB, serta kepala DB bantu (*auxiliary*) yang hanya aktif saat pelatihan [38].



Gambar 2.6. *Framework* PP-OCRv6 *Text Detection*

Sumber: [38]

Pada modul pengenalan teks, sebagaimana ditunjukkan pada Gambar 2.7, PP-OCRv6 menggunakan *backbone* LCNetV4 yang sama dengan *neck* LightSVTR, sebelum diumpankan ke kepala CTC untuk inferensi dan kepala NRTR untuk pelatihan [38].



Gambar 2.7. *Framework* PP-OCRv6 *Text Recognition*

Sumber: [38]

PP-OCRv6 tersedia dalam tiga tingkatan model yang berbagi primitif blok yang sama dengan konfigurasi kedalaman dan lebar yang berbeda [38]. Pertama, PP-OCRv6_medium dengan total 34,5 juta parameter merupakan model dengan kapasitas terbesar yang mencapai akurasi pengenalan 83,2% dan *detection H-mean* 86,2%, melampaui PP-OCRv5_server sebesar 5,1% dan 4,6% secara berurutan [38]. Kedua, PP-OCRv6_small dengan total 7,7 juta parameter menawarkan keseimbangan antara akurasi dan efisiensi, mencapai *detection H-mean* sebesar 84,1% sekaligus melampaui PP-OCRv5_server dengan empat kali lebih sedikit parameter [38]. Ketiga, PP-OCRv6_tiny dengan total 1,5 juta parameter merupakan model paling ringan yang dirancang untuk perangkat *edge*, mencapai kecepatan inferensi 3,9 kali lebih cepat dibandingkan PP-OCRv5_mobile pada CPU Intel Xeon dengan akurasi yang tetap kompetitif [38].

2.8 Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) merupakan metode pembobotan kata yang digunakan untuk mengubah data teks menjadi representasi numerik yang dapat diproses oleh algoritma *machine learning*. Metode ini menggabungkan dua komponen utama, yaitu *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF), untuk mengukur seberapa penting sebuah kata dalam suatu dokumen relatif terhadap keseluruhan koleksi dokumen [39].

Term Frequency (TF) mengukur seberapa sering sebuah kata atau istilah muncul di dalam suatu dokumen. Semakin sering sebuah kata muncul dalam dokumen tersebut, maka semakin besar nilai TF-nya. Nilai TF dihitung untuk setiap kata pada setiap dokumen secara individual [40]. Nilai *Term Frequency* dihitung

menggunakan Rumus 2.1.

$$TF(t,d) = f_{t,d} \tag{2.1}$$

dengan:

1. $TF(t,d)$ = nilai *Term Frequency* untuk kata t pada dokumen d
2. $f_{t,d}$ = frekuensi kemunculan kata t di dalam dokumen d

Pada beberapa implementasi, skema logaritmik digunakan untuk menormalisasi nilai TF agar tidak terlalu bias terhadap kata yang muncul sangat sering. Nilai *Term Frequency* berbasis logaritmik dihitung menggunakan Rumus 2.2.

$$TF(t,d) = 1 + \log(f_{t,d}) \tag{2.2}$$

Inverse Document Frequency (IDF) mengukur seberapa penting atau langka sebuah kata di seluruh koleksi dokumen. Kata-kata yang muncul di hampir semua dokumen, seperti kata penghubung, akan memiliki nilai IDF yang rendah karena dianggap kurang informatif. Sebaliknya, kata yang jarang muncul di banyak dokumen akan memiliki nilai IDF yang tinggi, menandakan kata tersebut bersifat lebih spesifik dan informatif [41]. Nilai *Inverse Document Frequency* dihitung menggunakan Rumus 2.3.

$$IDF(t) = \log\left(\frac{N}{df_t}\right) + 1 \tag{2.3}$$

dengan:

1. $IDF(t)$ = nilai *Inverse Document Frequency* untuk kata t
2. N = jumlah total dokumen dalam koleksi
3. df_t = jumlah dokumen yang mengandung kata t

TF-IDF merupakan hasil perkalian antara nilai TF dan IDF. Metode ini memberikan bobot yang lebih tinggi kepada kata-kata yang sering muncul dalam suatu dokumen tertentu namun jarang muncul di dokumen lainnya, sehingga kata

tersebut dianggap relevan dan representatif untuk dokumen tersebut [42]. Setelah teks diubah ke dalam bentuk vektor TF-IDF, algoritma klasifikasi seperti *Support Vector Machine* (SVM) dapat menggunakannya sebagai masukan untuk proses pelatihan dan pengujian model [39]. Nilai TF-IDF diperoleh dari hasil perkalian antara nilai TF dan IDF. Perhitungan TF-IDF ditunjukkan pada Rumus 2.4.

$$W(t,d) = TF(t,d) \times IDF(t) \tag{2.4}$$

dengan:

1. $W(t,d)$ = bobot TF-IDF untuk kata t pada dokumen d
2. $TF(t,d)$ = nilai *Term Frequency* kata t pada dokumen d
3. $IDF(t)$ = nilai *Inverse Document Frequency* kata t

Semakin tinggi nilai $W(t,d)$, maka semakin besar kontribusi kata t dalam merepresentasikan isi dokumen d dibandingkan dokumen-dokumen lain dalam koleksi.

2.9 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah algoritma pembelajaran mesin berbasis supervisi yang dirancang untuk menemukan *hyperplane* optimal pada ruang fitur berdimensi tinggi guna memisahkan data dari kelas yang berbeda dengan margin maksimum [41]. Titik-titik data yang berada paling dekat dengan *hyperplane* disebut *support vector*, dan jarak antara *support vector* dengan *hyperplane* disebut margin. Tujuan utama SVM adalah memaksimalkan margin tersebut sehingga model memiliki kemampuan generalisasi yang baik [40].

Secara formal, untuk dataset dengan n sampel (x_i, y_i), dengan $x_i \in \mathbb{R}^d$ merupakan vektor fitur dan $y_i \in -1, +1$ merupakan label kelas, SVM mencari *hyperplane* optimal yang ditunjukkan pada Rumus 2.5.

$$f(x) = w^T x + b = 0 \tag{2.5}$$

dengan w adalah vektor bobot (*weight vector*) dan b adalah nilai bias.

Proses klasifikasi pada SVM dilakukan berdasarkan tanda dari fungsi keputusan. Perhitungan prediksi kelas ditunjukkan pada Rumus 2.6.

$$\hat{y} = \text{sign}(w^T x + b) \tag{2.6}$$

dengan $\hat{y}$ merupakan hasil prediksi kelas untuk data masukan x.

Untuk memperoleh *hyperplane* yang optimal, SVM melakukan proses optimasi dengan meminimalkan norma vektor bobot. Fungsi kendala yang digunakan ditunjukkan pada Rumus 2.7.

$$y_i(w^T x_i + b) \geq 1, \quad \forall i \tag{2.7}$$

dengan x_i merupakan data ke-i, y_i merupakan label kelas data ke-i, sedangkan $\forall i$ menunjukkan bahwa kendala tersebut berlaku untuk seluruh data pelatihan.

Untuk data yang tidak dapat dipisahkan secara linear, SVM memanfaatkan fungsi *kernel* $K(x_i, x_j)$ yang berfungsi memetakan data ke ruang fitur berdimensi lebih tinggi tanpa perlu menghitung transformasi tersebut secara eksplisit [40]. Kernel yang paling sederhana adalah kernel linear yang dihitung menggunakan Rumus 2.8.

$$K(x_i, x_j) = x_i^T x_j \tag{2.8}$$

Selain kernel linear, terdapat kernel polinomial yang mampu menangkap hubungan non-linear antar data. Perhitungan kernel polinomial ditunjukkan pada Rumus 2.9.

$$K(x_i, x_j) = (\gamma \cdot x_i^T x_j + r)^d, \quad \gamma > 0 \tag{2.9}$$

Kernel lain yang banyak digunakan adalah *Radial Basis Function* (RBF) karena mampu menangani pola data yang kompleks. Rumus kernel RBF ditunjukkan pada Rumus 2.10.

$$K(x_i, x_j) = \exp\left(-\gamma \left| x_i - x_j \right|^2\right), \quad \gamma > 0 \tag{2.10}$$

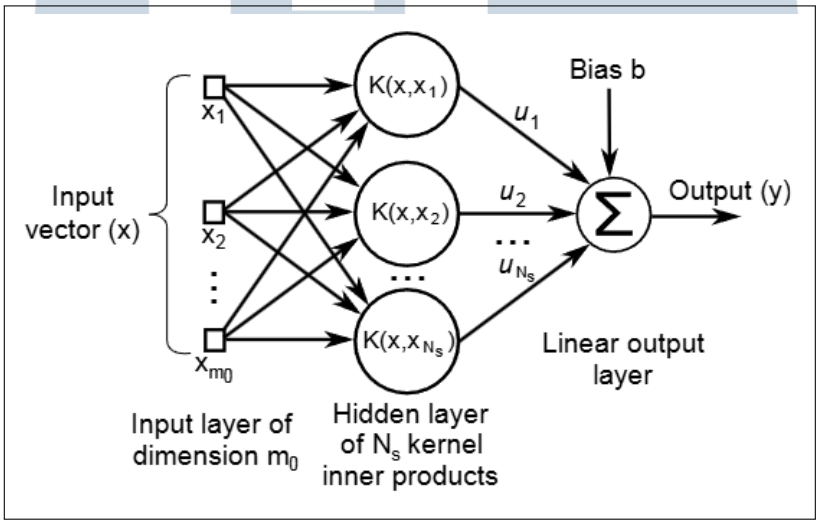
Selain itu, tersedia pula kernel sigmoid yang memiliki karakteristik

menyerupai fungsi aktivasi pada jaringan saraf tiruan. Perhitungan kernel sigmoid ditunjukkan pada Rumus 2.11.

$$K(x_i, x_j) = \tanh(\gamma \cdot x_i^T x_j + r) \tag{2.11}$$

Pada persamaan kernel di atas, γ, r, dan d merupakan parameter kernel yang dapat dikonfigurasi untuk menyesuaikan karakteristik data yang digunakan.

Arsitektur kerja SVM secara umum ditunjukkan pada Gambar 2.8.



Gambar 2.8. Arsitektur *Classifier* Berbasis SVM

Sumber: [43]

Secara garis besar, proses klasifikasi menggunakan SVM terdiri atas tiga tahapan utama, yaitu menentukan *hyperplane* sebagai batas pemisah antar kelas, memaksimalkan margin antara *hyperplane* dan *support vector*, serta mengelompokkan data ke dalam kelas berdasarkan posisi data terhadap *hyperplane* yang telah terbentuk [40].

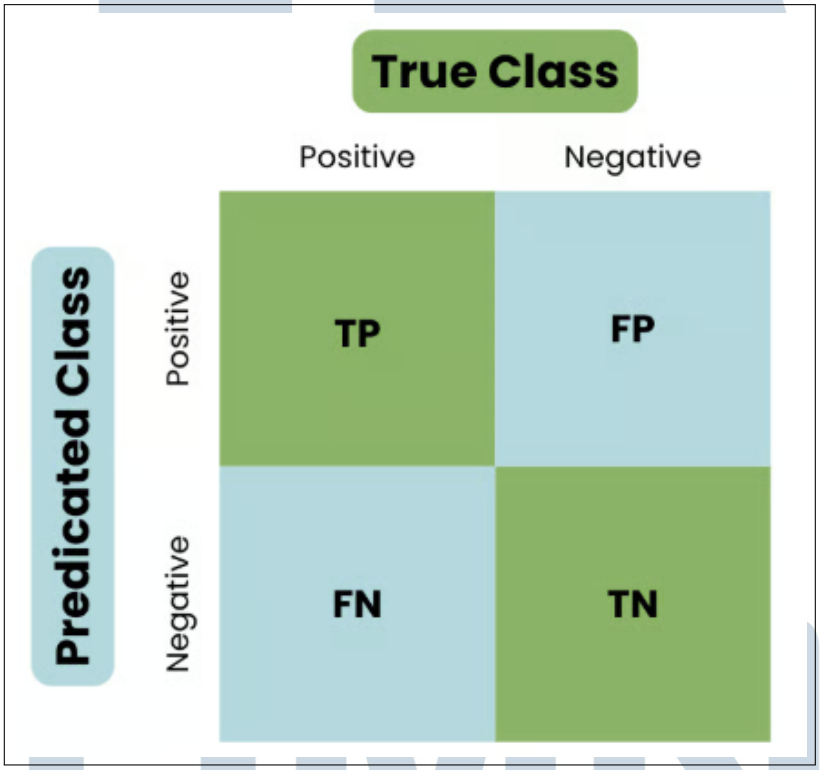
2.10 Evaluasi Model

2.10.1 Confusion Matrix

Confusion matrix adalah sebuah tabel evaluasi yang digunakan untuk mengukur kinerja model klasifikasi terawasi (*supervised classification*) dengan membandingkan hasil prediksi model terhadap label aktual [44]. Melalui *confusion*

matrix, performa model dapat dianalisis berdasarkan jumlah prediksi benar maupun prediksi salah pada masing-masing kelas. Selain itu, tabel ini juga menjadi dasar dalam perhitungan berbagai metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *F1-score*.

Secara umum, *confusion matrix* terdiri atas empat kombinasi hasil klasifikasi yang merepresentasikan hubungan antara kelas aktual dan kelas prediksi. Ilustrasi struktur *confusion matrix* dapat dilihat pada Gambar 2.9.



Gambar 2.9. Struktur *Confusion Matrix*
Sumber: [45]

Terdapat empat komponen utama dalam *confusion matrix*, yaitu sebagai berikut :

1. *True Positive* (TP) merupakan jumlah data pada kelas positif yang berhasil diprediksi dengan benar sebagai kelas positif oleh model.
2. *True Negative* (TN) merupakan jumlah data pada kelas negatif yang berhasil diprediksi dengan benar sebagai kelas negatif.
3. *False Positive* (FP) merupakan jumlah data pada kelas negatif yang salah

diprediksi sebagai kelas positif. Kondisi ini sering disebut sebagai *Type I Error*.

4. *False Negative* (FN) merupakan jumlah data pada kelas positif yang salah diprediksi sebagai kelas negatif. Kondisi ini dikenal sebagai *Type II Error*.

2.10.2 Accuracy

Accuracy merupakan pengukuran seberapa benar sebuah sistem dapat mengklasifikasikan data dari keseluruhan sampel [44]. Nilai *accuracy* menunjukkan proporsi prediksi yang benar, baik pada kelas positif maupun kelas negatif, terhadap seluruh data yang diuji. Rumus 2.12 menunjukkan cara perhitungan *Accuracy*.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.12}$$

2.10.3 Precision

Precision merupakan perbandingan antara jumlah data kategori positif yang berhasil diklasifikasikan secara benar oleh sistem dengan keseluruhan data yang diprediksi sebagai positif [44]. Nilai *precision* digunakan untuk mengukur ketepatan sistem dalam memberikan prediksi positif. Rumus 2.13 menunjukkan cara perhitungan *Precision*.

$$Precision = \frac{TP}{TP + FP} \tag{2.13}$$

2.10.4 Recall

Recall merupakan pengukuran terhadap kemampuan sistem dalam mendeteksi seluruh data positif secara benar [44]. Nilai *recall* menunjukkan seberapa banyak data positif aktual yang berhasil dikenali oleh model. Rumus 2.14 menunjukkan cara perhitungan *Recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.14}$$

2.10.5 F1-Score

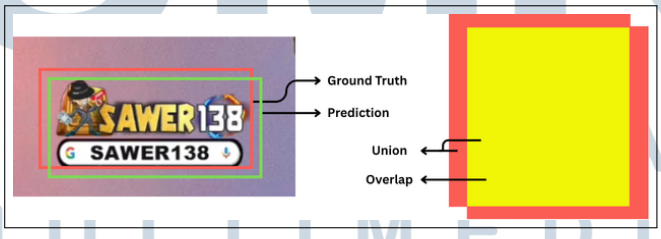
F1-Score merupakan metrik evaluasi yang menggabungkan nilai *precision* dan *recall* menggunakan *harmonic mean* [44]. Metrik ini digunakan untuk memberikan ukuran performa yang lebih seimbang, terutama ketika distribusi data tidak seimbang. Rumus 2.15 menunjukkan cara perhitungan *F1-score*.

$$F\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.15}$$

2.10.6 Intersection over Union (IoU)

Intersection over Union (IoU) merupakan metrik evaluasi yang digunakan untuk mengukur tingkat tumpang tindih antara *bounding box* hasil prediksi model dengan *bounding box* anotasi sebenarnya (*ground truth*) [46]. Nilai IoU digunakan sebagai ambang batas (*threshold*) untuk menentukan apakah suatu hasil prediksi dikategorikan sebagai *True Positive* (TP), *False Positive* (FP), atau *False Negative* (FN). Semakin besar nilai IoU, semakin tinggi tingkat kesesuaian lokasi objek yang diprediksi model terhadap posisi objek sebenarnya.

Gambar 2.10 menunjukkan ilustrasi perhitungan *Intersection over Union* yang diperoleh dari perbandingan antara area prediksi model dengan area anotasi sebenarnya (*ground truth*). Area yang saling bertumpang tindih disebut sebagai *intersection*, sedangkan total gabungan kedua area tersebut disebut sebagai *union*.



Gambar 2.10. Ilustrasi Perhitungan *Intersection over Union* (IoU)
Sumber: [46]

Rumus 2.16 menunjukkan cara perhitungan *Intersection over Union*.

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \tag{2.16}$$

Berdasarkan Rumus 2.16, nilai IoU diperoleh dengan membagi luas area irisan (*Area of Intersection)*) dengan luas total gabungan area prediksi dan area *ground truth* (*Area of Union*). Semakin besar area irisan dan semakin kecil area yang tidak saling bertumpang tindih, maka nilai IoU akan semakin tinggi.

Nilai IoU berada pada rentang 0 hingga 1. Nilai 0 menunjukkan tidak adanya tumpang tindih antara area prediksi dan *ground truth*, sedangkan nilai 1 menunjukkan tumpang tindih sempurna. Semakin mendekati nilai 1, semakin akurat posisi *bounding box* hasil prediksi terhadap objek sebenarnya.

Dalam pelatihan maupun evaluasi model deteksi objek seperti YOLO, nilai IoU digunakan sebagai parameter penting dalam menentukan kualitas prediksi *bounding box*. Ambang batas yang umum digunakan adalah IoU ≥ 0.5 pada evaluasi *mAP50*, sedangkan pada metrik *mAP50–95*, evaluasi dilakukan pada rentang threshold IoU 0.5 hingga 0.95 dengan interval 0.05.

2.10.7 Mean Average Precision (mAP)

Mean Average Precision (mAP) merupakan metrik evaluasi standar yang umum digunakan untuk mengukur performa model deteksi objek. Metrik ini merangkum kemampuan model dalam mendeteksi objek pada seluruh kelas dengan mempertimbangkan berbagai nilai ambang *Intersection over Union* (IoU) [47]. Perhitungan mAP diawali dengan menghitung nilai *Average Precision* (AP) untuk setiap kelas objek. Nilai AP diperoleh dari luas area di bawah kurva *precision-recall*, yang merepresentasikan hubungan antara nilai *precision* dan *recall* pada berbagai threshold prediksi.

Rumus 2.17 menunjukkan cara perhitungan *Mean Average Precision*.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.17}$$

Berdasarkan Rumus 2.17, nilai mAP diperoleh dengan menghitung rata-rata nilai *average precision* dari seluruh kelas objek yang dievaluasi, dengan N merupakan jumlah kelas objek dan AP_i merupakan nilai *average precision* pada kelas ke-i. Semakin tinggi nilai mAP, semakin baik kemampuan model dalam mendeteksi objek secara akurat baik dari sisi klasifikasi maupun lokalisasi posisi objek.

Dalam evaluasi model YOLO, nilai mAP umumnya diukur pada threshold IoU tertentu, seperti *mAP50* yang menggunakan threshold IoU sebesar 0,5, serta *mAP50–95* yang menghitung rata-rata mAP pada rentang threshold IoU 0,5 hingga 0,95.

2.10.8 Character Error Rate (CER)

Character Error Rate (CER) merupakan metrik evaluasi standar yang digunakan untuk mengukur tingkat kesalahan sistem pengenalan teks pada level karakter. Dalam konteks *Optical Character Recognition* (OCR), CER dihitung berdasarkan *edit distance* atau jarak Levenshtein antara teks hasil ekstraksi sistem dengan teks referensi (*ground truth*). Perhitungan tersebut mempertimbangkan tiga jenis operasi edit, yaitu substitusi (*substitution*), penghapusan (*deletion*), dan penyisipan (*insertion*) karakter. Semakin kecil nilai CER, semakin baik performa sistem OCR dalam mengenali karakter secara akurat [48, 49].

Rumus 2.18 menunjukkan cara perhitungan *Character Error Rate*.

$$CER = \frac{S + D + I}{N} \tag{2.18}$$

Berdasarkan Rumus 2.18, S menyatakan jumlah karakter yang mengalami kesalahan substitusi (*substitution*), yaitu karakter yang dikenali sebagai karakter lain; D menyatakan jumlah karakter yang hilang atau tidak berhasil dikenali (*deletion*); I menyatakan jumlah karakter tambahan yang tidak terdapat pada teks referensi (*insertion*); sedangkan N merupakan jumlah total karakter pada teks *ground truth*.

Nilai CER berada pada rentang 0 hingga lebih dari 0, di mana nilai CER = 0 menunjukkan hasil pengenalan karakter yang sempurna tanpa kesalahan. Semakin rendah nilai CER, semakin tinggi tingkat akurasi sistem OCR dalam mengekstraksi teks. CER digunakan untuk mengevaluasi kemampuan sistem OCR dalam mengenali karakter pada watermark promosi judi online, khususnya pada teks pendek berupa kata, frasa, atau kombinasi huruf dan angka yang sering muncul pada watermark digital.

2.10.9 Word Error Rate (WER)

Word Error Rate (WER) merupakan metrik evaluasi yang digunakan untuk mengukur tingkat kesalahan sistem pengenalan teks pada level kata. Dalam konteks *Optical Character Recognition* (OCR), WER dihitung dengan membandingkan teks hasil ekstraksi sistem dengan teks referensi (*ground truth*) berdasarkan jumlah kata yang mengalami substitusi, penghapusan, maupun penyisipan. Metrik ini memberikan gambaran mengenai seberapa akurat sistem dalam mengenali susunan kata secara keseluruhan. Semakin rendah nilai WER, semakin baik performa sistem OCR dalam menghasilkan keluaran teks yang sesuai dengan *ground truth* [48, 49].

Rumus 2.19 menunjukkan cara perhitungan *Word Error Rate*.

$$WER = \frac{S_w + D_w + I_w}{N_w} \tag{2.19}$$

Berdasarkan Rumus 2.19, S_w menyatakan jumlah kata yang mengalami kesalahan substitusi (*substitution*), yaitu kata yang dikenali sebagai kata lain; D_w menyatakan jumlah kata yang terhapus atau tidak berhasil dikenali (*deletion*); I_w menyatakan jumlah kata tambahan yang tidak terdapat pada *ground truth* (*insertion*); dan N_w merupakan jumlah total kata pada teks referensi (*ground truth*).

Nilai WER berada pada rentang 0 hingga lebih dari 0, di mana nilai WER = 0 menunjukkan hasil pengenalan kata yang sempurna tanpa kesalahan. Semakin rendah nilai WER, semakin tinggi tingkat akurasi sistem OCR dalam mengekstraksi teks. WER digunakan untuk mengevaluasi kemampuan sistem OCR dalam mengenali kata pada watermark promosi judi online, khususnya pada teks pendek berupa kata, frasa, atau kombinasi huruf dan angka yang sering muncul pada watermark digital.