BAB 2
LANDASAN TEORI

2.1 Citra Digital (Digital Image)

Citra digital adalah matriks 2 dimensi yang merupakan proyeksi lingkungan 3 dimensi ke dalam gambar 2 dimensi menggunakan pengukuran numerikal dari *picture element* atau *pixel* [19, 20]. Setiap *pixel* merepresentasikan lokasi dan intensitas dari setiap poin pada gambar [19]. Citra digital dapat direpresentasikan dalam berbagai bentuk seperti *color* ataupun *grayscale*. Sebuah *color image* memiliki 3 representasi *pixel*, yaitu *red* (R), *green* (G), dan *blue* (B), sedangkan *grayscale image* hanya memiliki 1 representasi pixel. Setiap pixel memiliki intensitasnya sendiri yang berkisar dari nilai 0 sampai dengan 255 [19]. Intensitas ini menunjukkan nilai dari warna pada setiap *pixel* yang ada dengan 0 sebagai paling hitam dan 255 sebagai paling putih pada *grayscale image* [20]. Hal ini juga berlaku sama pada *color image*. Contohnya pada *red* (R), 0 melambangkan paling hitam dan 255 sebagai paling merah.

2.2 Convolutional Neural Network (CNN)

Convolutional neural network (CNN) merupakan salah satu cabang *deep learning* yang pada awalnya dikhususkan untuk melakukan *image processing* [21]. Arsitektur CNN terdiri dari *convolution layers* untuk melakukan ekstraksi fitur dan *pooling layers* untuk melakukan *sampling* dari fitur hasil ekstraksi *convolution layers* yang diakhiri dengan *fully connected layers* [21, 22]. *Convolution layers* menggunakan metode penyaringan pada gambar yang nantinya akan dilanjutkan *pooling layers* untuk mengurangi ukuran dari gambar [21]. CNN banyak digunakan dalam citra digital, khususnya dalam melakukan *image classification, object detection*, ataupun pada kategori seperti *speech recognition*, dan *natural language processing* (NLP). [22, 21].

2.3 Region Based Convolutional Neural Network (R-CNN)

Region based convolutional network (R-CNN) merupakan sebuah *two-stage object detection architecture* yang menggunakan *region proposal network* (RPN) untuk menentukan prediksi *region* obyek pada gambar, yang dilanjutkan dengan

tahap klasifikasi obyek serta regresi *bounding box* yang dilakukan dengan CNN [11, 23]. Kekurangan dari R-CNN, obyek yang dilakukan prediksi bisa memiliki lokasi *region* yang berbeda-beda dengan ukuran *ratio* yang berbeda juga. Ditambah lagi, tahap klasifikasi dan regresi yang membuat komputasi menjadi sangat berat.

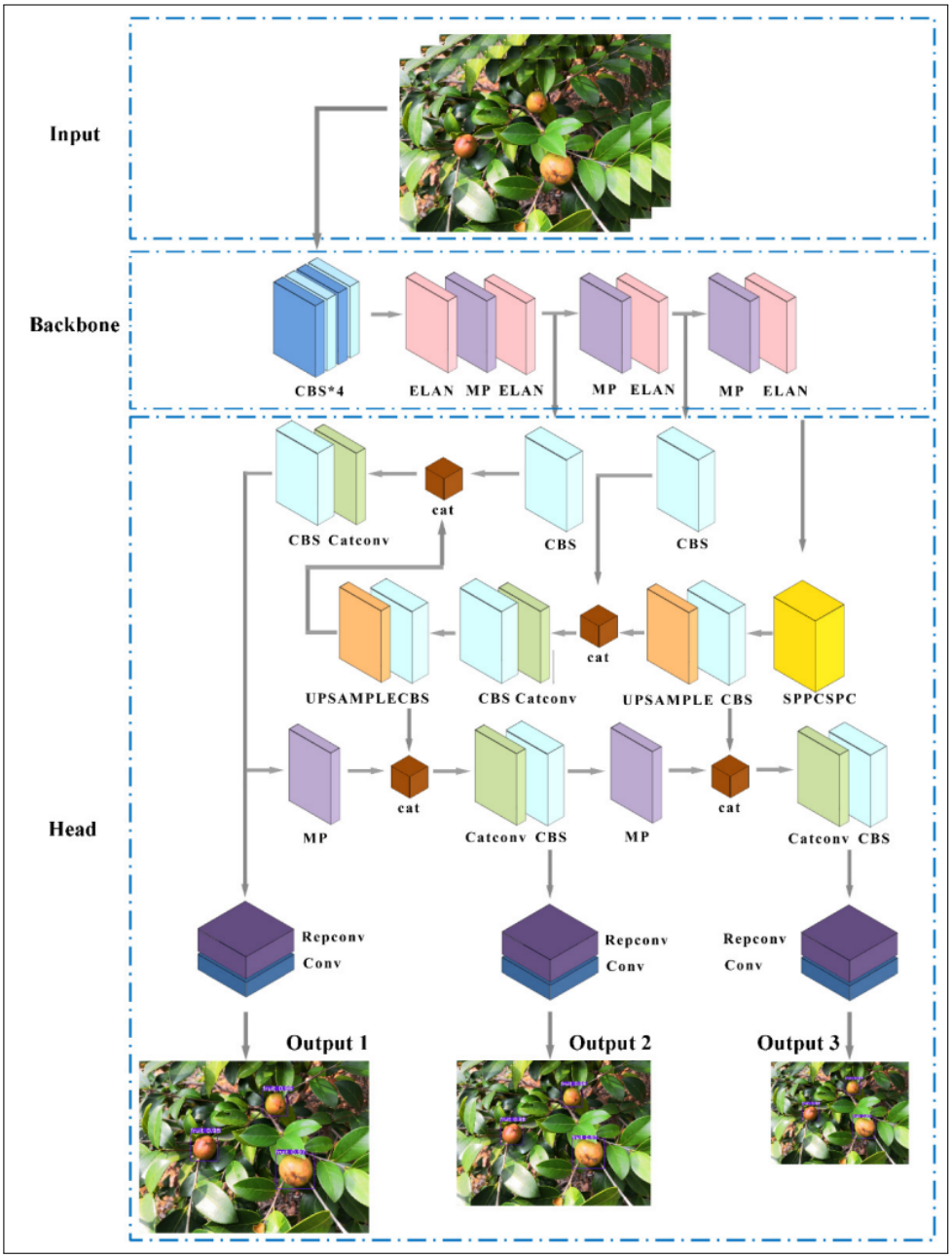
2.4 You Only Look Once (YOLO)

You only look once (YOLO) merupakan sebuah *single-stage object detection architecture* yang langsung mengirimkan gambar ke *layer* CNN untuk dilakukan prediksi *bounding box* beserta dengan obyek pada setiap *box* tersebut [11, 24]. YOLO menggunakan metode regresi untuk mendeteksi target obyek dan langsung mendeteksi gambar secara global sehingga mengurangi kemungkinan deteksi *background* sebagai obyek [25]. Oleh karena hal ini, YOLO memiliki performa yang lebih cepat dibandingkan R-CNN terutama dalam hal kecepatan *inference* [25].

2.5 YOLOv7

YOLOv7 merupakan iterasi ke-7 YOLO dengan arsitektur yang terbagi menjadi input, *backbone*, dan *head* [26, 11].

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

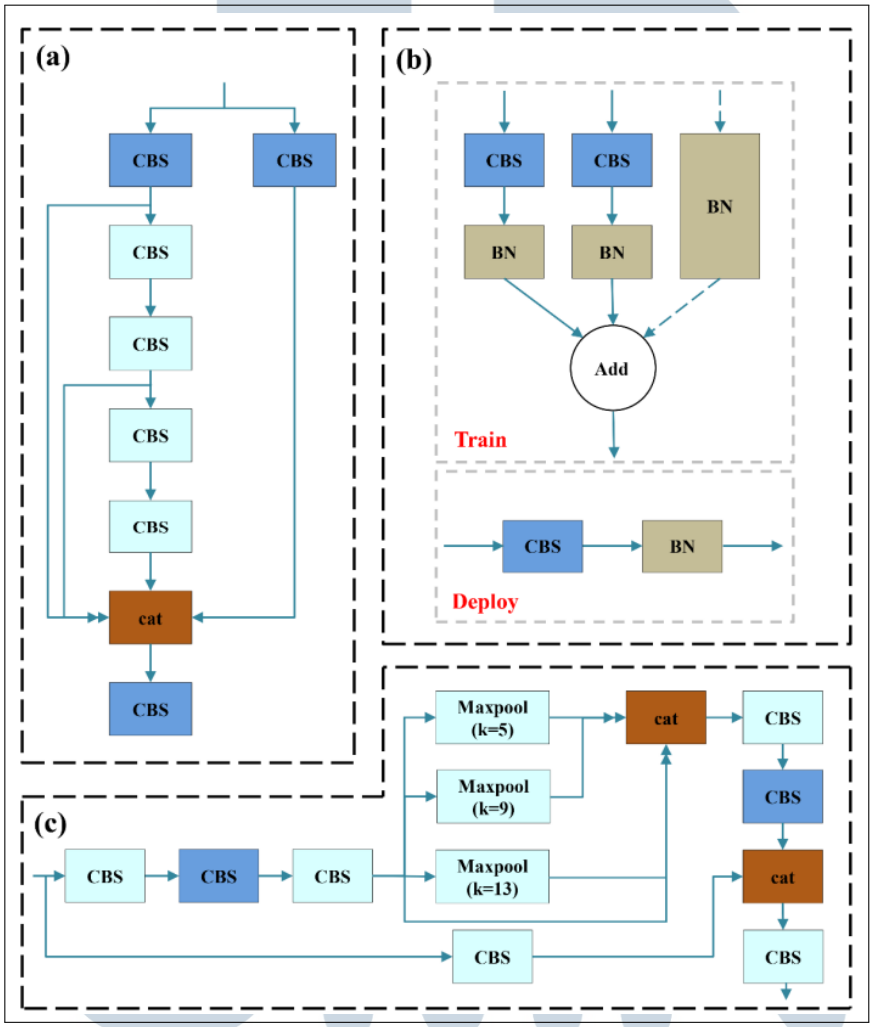


Gambar 2.1. YOLOv7 Network Architecture

UNIVERSITAS MULTIMEDIA NUSANTARA

Dapat dilihat pada Gambar 2.1, pertama-tama input gambar dilakukan *pre-processing* dan diubah menjadi gambar dengan *ratio* 640x640x3 [26]. Selanjutnya, gambar hasil *pre-processing* dimasukkan ke dalam *backbone layer* yang menggunakan arsitektur E-ELAN [26]. Di *backbone layer*, modul CBS *composite*, ELAN, dan MP secara bergantian mengurangi *width* dan *ratio* dari *feature map* sebanyak setengah serta menaikkan jumlah *output channel* menjadi 2 kali *input channel* [26]. Pada modul CBS *composite*, dilakukan *convolution*,

batch normalization (BN), dan *activation function* SiLu pada *input feature map* [9]. Modul ELAN terdiri dari banyak modul CBS yang digabungkan menjadi satu sehingga akan lebih banyak fitur yang dapat dipelajari dengan lebih baik [9]. Modul MP terdiri dari beberapa modul CBS dan 1 modul *MaxPool* [9].

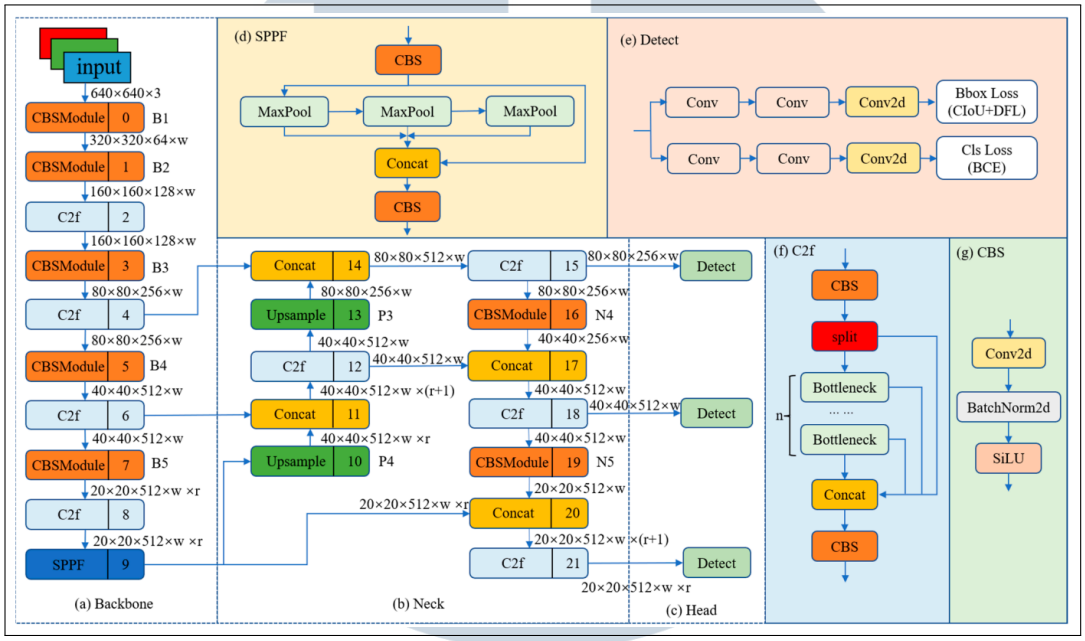


Gambar 2.2. Struktur dari setiap modul YOLOv7. (a) ELAN; (b) Repconv; (c) SPPCSPC

Pada *head layer*, hasil *output* dari *backbone* dilakukan prediksi obyek, *class*, dan *bounding box* dari setiap obyek [26]. Bagian *head* terdiri dari modul SPPCSPC, CBS, MP, Catconv, serta 3 Repconv sebelum dihasilkan 3 *feature map output* [26]. Modul SPPCSPC bertujuan untuk meningkatkan bagian reseptif dari *network* untuk memperkaya extraksi fitur [9, 26]. Modul ELAN, Repconv, dan SPPCSPC dapat dilihat dengan lebih jelas pada Gambar 2.2.

2.6 YOLOv8

YOLOv8 merupakan iterasi ke-8 YOLO dengan arsitektur yang terbagi menjadi *backbone*, *neck*, dan *head* [13, 12].



Gambar 2.3. YOLOv8 Network Architecture

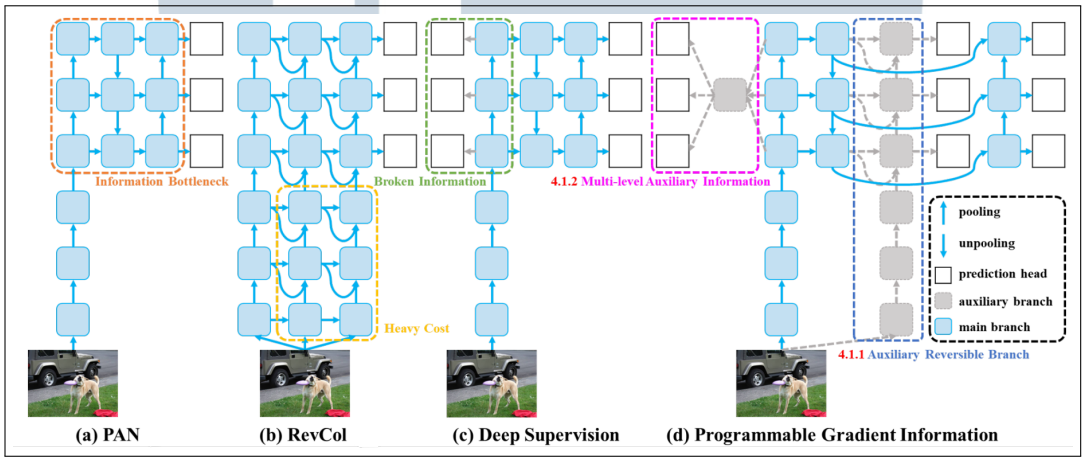
Dapat dilihat pada Gambar 2.3, pertama-tama input gambar dikirim ke *backbone* untuk dilakukan *down sampling* sebanyak 5 kali melalui modul CBS yang menghasilkan 5 fitur dengan skala berbeda-beda [13]. Selain modul CBS, terdapat modul C2f sebagai modul CSP dalam memperkaya informasi dari ekstraksi fitur sambil mempertahankan keringanan komputasi [13]. Modul CBS melakukan *convolution* pada input, diikuti dengan *batch normalization* (BN), dan *activation function* SiLu untuk mengaktifkan *stream* dari informasi [13]. Setelah itu, *backbone layer* diakhiri dengan modul *spatial pyramid pooling fast* (SPPF) yang melakukan *pooling input feature maps* ke dalam *map* dengan ukuran tetap untuk *output* dengan ukuran yang adaptif [13].

Pada bagian *neck*, YOLOV8 menggunakan struktur PAN-FPN yang menghilangkan proses *convolution* setelah *up-sampling* seperti pada YOLOV5 dan YOLOV7 [13]. stuktur konvensional FPN menggunakan pendekatan *top-down* yang menyebabkan lokalisasi obyek yang buruk [13]. PAN-FPN menggunakan pendekatan *top-down* dan *bottom-up* yang dapat menyadari lokasi informasi yang dangkal dan informasi semantik yang dalam melalui penggabungan fitur [13].

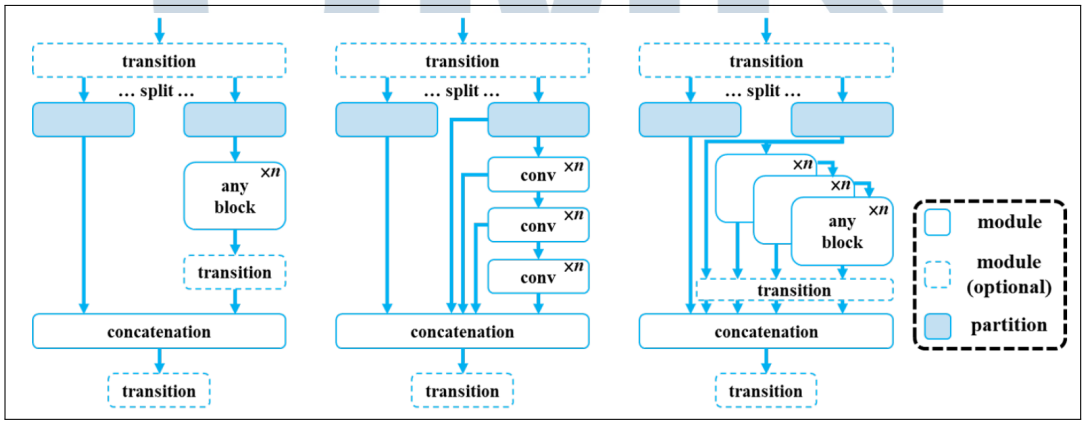
Pada bagian *head*, YOLOV8 menggunakan struktur *head* terpisah yang terbagi menjadi 2 cabang, yaitu untuk klasifikasi obyek dan regresi prediksi *bounding box* [13]. Untuk klasifikasi obyek, *binary cross-entropy loss* (BCE loss) digunakan sebagai *loss function*, sedangkan untuk regresi prediksi *bounding box*, *distribution focal loss* (DFL) dan CIoU digunakan sebagai *loss function* [13].

2.7 YOLOv9

YOLOv9 merupakan iterasi ke-9 YOLO dengan arsitektur yang terbagi menjadi *backbone* dan *neck* [14].



Gambar 2.4. YOLOv9 PGI Network Architecture. (a) path aggregation network (PAN), (b) reversible columns (RevCol), (c) conventional deep supervision, (d) programmable gradient information (PGI)



Gambar 2.5. YOLOv9 GELAN Network Architecture. (a) CSPNet, (b) ELAN, (c) GELAN

YOLOv9 menggunakan arsitektur yang terdiri dari gabungan *programmable*

gradient information (PGI) dan *generalized efficient layer aggregation network* (GELAN) yang dapat dilihat pada Gambar 2.4 dan 2.5 [14]. PGI digunakan untuk mengurangi *information bottleneck* yang dapat menyebabkan *loss function* menghasilkan nilai gradien yang kurang bagus [14]. PGI sendiri terdiri dari 3 komponen utama, yaitu *main branch*, *auxiliary reversible branch*, dan *multi-level auxiliary information* seperti yang bisa dilihat pada gambar 2.4 [14].

Auxiliary reversible branch bertugas dalam mengurangi kemungkinan *loss function* menemukan korelasi yang tidak relevan dengan target dari fitur *feedforward* yang masih belum selesai [14]. Hal ini dilakukan dengan melakukan *mapping* informasi dari data ke target. Dengan ini, fitur-fitur *main branch* mendapatkan informasi gradien yang lebih berguna dalam mencegah terjadinya *information bottleneck* [14]. *Multi-level auxiliary information* bertugas dalam mencegah terjadinya dominansi informasi tertentu pada obyek sehingga prediksi dapat dilakukan pada informasi yang lebih luas dan adaptif [14]. Hal ini dilakukan dengan mengintegrasikan sebuah network di antara fitur piramid dari *auxiliary supervision* dan *main branch*. Nantinya, gradien dari berbagai *prediction head* akan digabungkan dan dikirim ke *main branch* untuk dilakukan pembaharuan parameter lebih lanjut. GELAN seperti yang bisa dilihat pada gambar 2.5 merupakan sebuah arsitektur gabungan dari CSPNet dan ELAN [14]. ELAN menggunakan blok *convolutional* yang ditumpuk menjadi beberapa lapisan, sedangkan CSPNet dapat menggunakan blok komputasi apapun yang diikuti dengan blok transisi. GELAN menggunakan komputasi blok apapun yang ditumpuk serta diikuti oleh blok transisi. Dengan pendekatan ini, didapatkan arsitektur yang lebih *lightweight* dibandingkan dengan pendekatan sebelumnya [14].

2.8 Confusion Matrix

Confusion matrix merupakan salah satu metode evaluasi yang sering digunakan dalam *machine learning* terutama untuk *supervised classification* dari model yang dapat diklasifikasi [27]. Dalam confusion matrix, terdapat 2 jenis data, yaitu data aktual dan data prediksi. Data aktual merupakan data yang telah dibuktikan kebenarannya dan akan dijadikan sebagai dasar dalam mengevaluasi data prediksi, sedangkan data prediksi merupakan data hasil prediksi dari algoritma yang digunakan [27]. Jika diproyeksikan ke dalam tabel seperti pada Tabel 2.1, baris tabel merepresentasikan kelas data aktual dan kolom tabel merepresentasikan kelas data prediksi. Biasanya, tabel akan berukuran 2x2 yang memiliki 4 ukuran

yaitu seperti berikut.

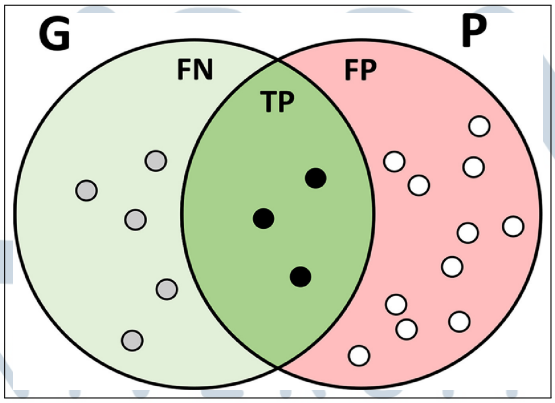
1. ”True Positive” (TP) - Data Prediksi *Positive* dan benar
2. ”True Negative” (TN) - Data Prediksi *Negative* dan benar
3. ”False Positive” (FP) - Data Prediksi *Positive* tetapi salah
4. ”False Negative” (FN) - Data Prediksi *Negative* tetapi salah

	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Tabel 2.1. Tabel Confusion Matrix

2.9 Precision and Recall

Precision dan *recall* merupakan ukuran yang paling sering digunakan dalam bidang *pattern recognition* untuk mengevaluasi hasil informasi yang didapatkan [28]. Perbedaan *precision* dan *recall* dapat dilihat dengan lebih jelas pada Gambar 2.6.



Gambar 2.6. Ground truth (G) dan predicted result (P)

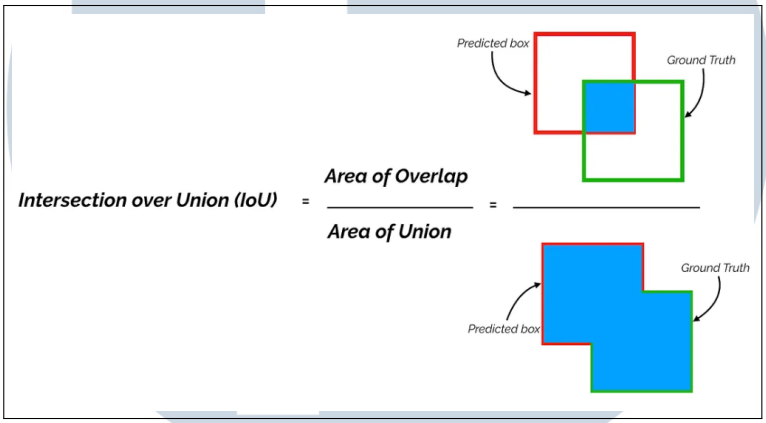
Precision merupakan jumlah dari TP dari keseluruhan jumlah hasil yang didapat, sedangkan *recall* merupakan jumlah dari TP dari keseluruhan jumlah hasil yang diekspektasi [28].

$$Precision = \frac{TP}{TP+FP} = \frac{G \cap P}{P} \tag{2.1}$$

$$Recall = \frac{TP}{TP+FN} = \frac{G \cap P}{G} \tag{2.2}$$

2.10 Intersection Over Union (IoU)

Intersection over union (IoU) merupakan sebuah ukuran evaluasi yang digunakan untuk mengukur akurasi dari *object detector* yang telah dilakukan *training* dari *dataset* [29].



Gambar 2.7. Intersection over union

Untuk menentukan IoU diperlukan *bounding box* yang didapat dari *dataset* awal yang telah diberi label (*ground-truth*), serta *bounding box* hasil prediksi dari model (*predicted*) [29]. IoU dapat dihitung dengan membagi luas area dari *ground-truth* yang bertumpang tindih dengan *predicted* dengan luas area total dari keduanya seperti pada Gambar 2.7 [29]. Nilai IoU ini dapat digunakan untuk menentukan TP dan FP dari sebuah prediksi berdasarkan *threshold* yang ditentukan. Bila IoU prediksi melebihi dari *threshold* IoU, maka prediksi dinyatakan TP, sedangkan bila IoU prediksi kurang dari *threshold* IoU, maka prediksi dinyatakan FP [30].

2.11 Mean Average Precision (mAP)

Mean average precision (mAP) merupakan salah satu ukuran evaluasi yang cukup terkenal dalam mengukur akurasi *object detection model* [30]. Akurasi mAP dihitung dengan mengambil rata-rata dari *average precision* (AP) dari setiap *class* menggunakan *threshold* IoU yang ditentukan [30, 9].

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{2.3}$$