BAB 2
LANDASAN TEORI

2.1 Deep Learning

Deep Learning (DL) merupakan bagian dari pembelajaran mesin yang berdasar pada tingkat representasi pembelajaran, sesuai dengan hierarki fitur. Dimana fitur *high-level* didapatkan dari fitur *low-level*, dan fitur *low-level* akan membantu mendefinisikan fitur-fitur yang lebih tinggi (*higher level*)[6]. *Deep Learning* menambah jumlah *Hidden Layers* diantara lapisan *input* dan lapisan *output* pada *Neural Networks* konvensional. Penambahan jumlah *Hidden Layers* dilakukan untuk menggambarkan model hubungan yang bersifat kompleks dan nonlinier. Performa yang baik dari DL selama beberapa tahun terakhir, menyebabkan DL banyak digunakan untuk analisis citra seperti segmentasi dan klasifikasi [7].

2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah sebuah arsitektur *Deep Learning* yang seringkali digunakan untuk mengolah citra gambar, segmentasi, *synthetic image generation*, dan klasifikasi gambar. Pada awalnya CNN akan mengekstraksi fitur *low-level* dari *input* mentah, kemudian CNN akan mengekstrasi fitur *high-level* pada lapisan yang lebih dalam. Melalui tahapan-tahapan ini, CNN dapat melakukan deteksi fitur visual (bentuk sederhana, tepi gambar, dan deteksi objek secara utuh atau lengkap) yang kompleks dari sebuah citra[8].

Secara historis, CNN memiliki rekam jejak yang baik untuk implementasi dalam hal analisa dan interpretasi citra [3]. CNN dapat mengklasifikasikan 1.2 juta gambar kedalam 1000 kelas, dengan tingkat kesalahan yang lebih rendah dibandingkan dengan model *Deep Learning* yang lain[9]. Arsitektur CNN terdiri dari 2 bagian, yaitu *Feature Extraction Layer* dan *Fully-Connected Layer* (MLP), dua bagian inilah yang membedakan CNN dari *Neural Network* yang lain. *Feature Extraction Layer* terdiri dari *yaitu convolutional layer* dan *pooling layer*. Kelebihan dari CNN adalah CNN cocok digunakan untuk pembelajaran menggunakan data mentah tanpa memilih fitur apriori dan memiliki performa yang relatif baik untuk dataset yang berukuran besar[10].

2.2.1 Convolutional Layer

Model CNN dapat memiliki lebih dari satu lapisan konvolusional. Pada umumnya, lapisan konvolusional yang pertama bertugas untuk mengekstrak fitur *low-level* seperti warna, tepian, dan gradien. Seiring bertambahnya lapisan pada model, model dapat beradaptasi untuk mengekstrak fitur yang bersifat *high-level*, sehingga jaringan pada pada model dapat memahami citra yang ada pada *dataset* dengan baik[11].

2.2.2 Pooling Layer

Pooling Layer bertujuan untuk mengurangi dimensi fitur konvolusi atau *feature map*, sehingga mengurangi daya komputasi yang diperlukan untuk memproses fitur atau data, karena semakin sedikit variabel yang harus diperbarui dan sekaligus mengatasi *overfitting*. Pada *Pooling Layer* terdapat 2 jenis *pooling*, yaitu *Max Pooling* dan *Average Pooling*. *Max Pooling* mengembalikan sebuah nilai maksimal dari bagian citra yang dicakup oleh *kernel* (Elemen yang terlibat dalam menjalankan operasi konvolusi pada bagian pertama di *convolutional layer*). Sedangkan *Average Pooling* mengembalikan nilai rata-rata dari nilai-nilai pada bagian citra yang dicakup oleh *kernel*. Jumlah *Convolutional Layer* dan *Pooling Layer* dapat ditingkatkan sesuai dengan kompleksitas citra, agar bisa menangkap lebih banyak detail fitur *low-level* [12].

2.2.3 Fully Connected Layer

Fully Connected Layer adalah *Multi Layer Perceptron* (MLP) yang terdiri dari *beberapa hidden layer, activation function, output layer,* dan *loss function*. Setelah citra diproses oleh *Convolutional Layer* dan *Pooling Layer*, model CNN sudah berhasil diaktifkan untuk memahami fitur-fitur yang ada pada citra. Kemudian, fitur yang dihasilkan akan menjadi *input* untuk *Neural network* untuk diklasifikasi. Penambahan *Fully Connected Layer* merupakan metode yang umum digunakan agar model dapat mempelajari kombinasi non-linier dari fitur *high-level* yang sudah didapat.

2.3 YOLOv7

You Only Look Once (YOLO) merupakan sebuah algortima *Object Detection* yang terkenal karena kecepatan dan akurasi yang dimilikinya. YOLO pertama kali diperkenalkan pada tahun 2015 dan saat ini sudah mencapai iterasi ke-7 (YOLOv7). YOLOv7 pertama kali diperkenalkan oleh Chien-Yao Wang, Alexey Bochkovskiy, dan Hong-Yuan Mark Liao pada tahun 2022 melalui paper berjudul "*YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*". YOLOv7 berfokus pada beberapa metode optimasi untuk meningkatkan akurasi [13].

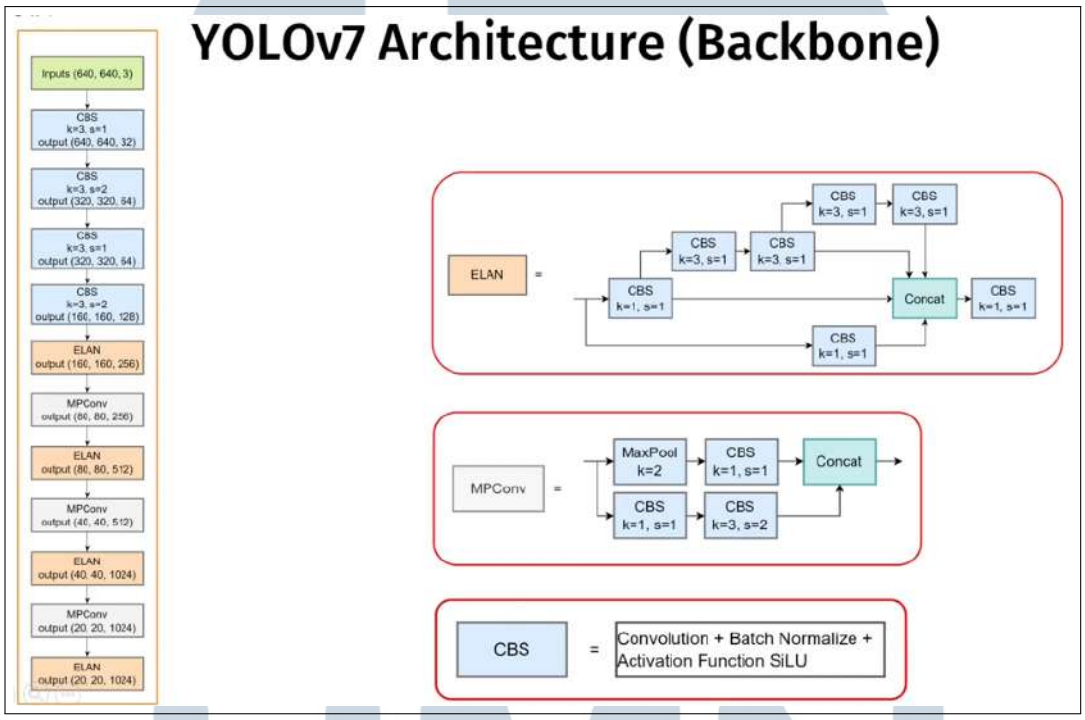
2.3.1 Arsitektur YOLOv7

Sebagai *single stage object detector*, YOLOv7 mengevaluasi sebuah gambar secara keseluruhan dalam satu langkah. Untuk setiap objek yang ada pada gambar, YOLOv7 memprediksi beberapa *bounding box* dan kemungkinan kelas untuk setiap *bounding box*. Secara garis besar arsitektur YOLOv7 terdiri dari 3 bagian, yaitu *backbone, neck,* dan *head*[13]. Penjelasan mengenai setiap bagian adalah sebagai berikut.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

A Flowchart YOLOv7 Bagian Backbone

Bagian *Backbone* memiliki tanggung jawab untuk menerima input gambar atau video lalu melakukan *resizing* pada gambar input, kemudian melakukan ektraksi fitur esensial dari input. Oleh karena itu, bagian *Backbone* banyak diisi dengan lapisan konvolusi seperti pada Gambar 2.1 berikut.

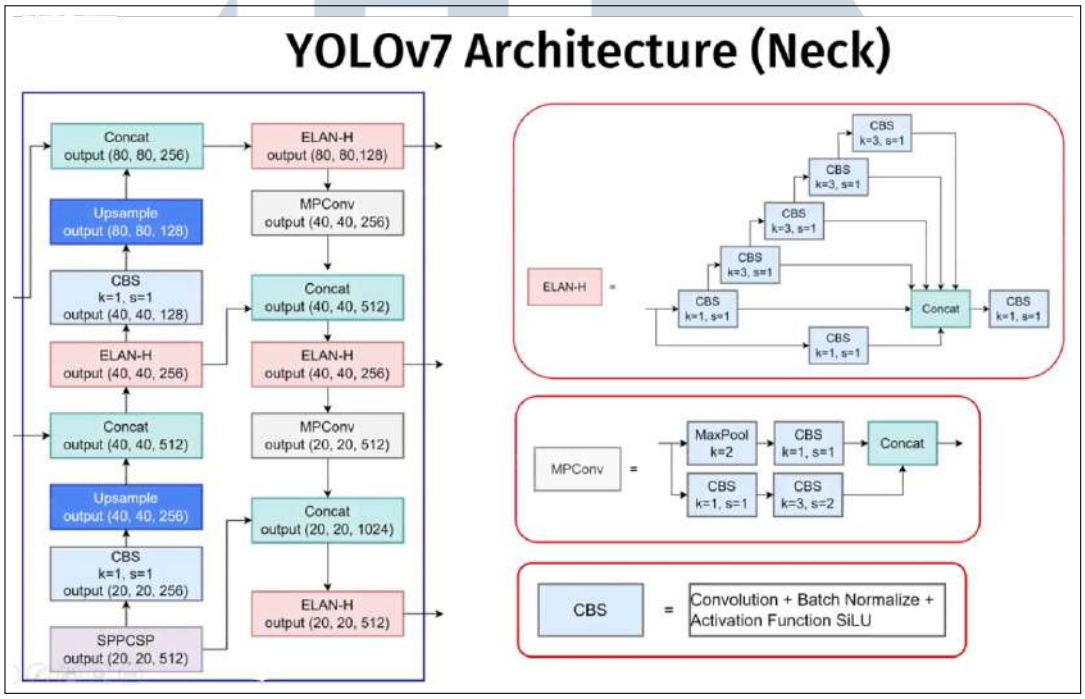


Gambar 2.1. *Flowchart Backbone* YOLOv7

Pada Gambar 2.1 bagian *Backbone* YOLOv7 terdiri dari CBS, ELAN, dan MPConv. CBS merupakan gabungan dari *convolution, Batch Normalization*, dan fungsi aktivasi SiLU. CBS menghubungkan lapisan konvolusi dengan *Batch Normalization* secara langsung, dengan tujuan untuk mengintegrasikan *mean* dan *variance* dari *Batch Normalization* kepada bias dan *weight* dari lapisan konvolusi. Lapisan MPConv pada YOLOv7 terdiri dari beberapa blok CBS dan sebuah blok *MaxPool*. ELAN pada YOLOv7 terdiri dari blok-blok CBS yang pada akhirnya digabungkan(*concatenate*). ELAN digunakan agar model dapat mempelajari fitur lebih banyak dengan lebih baik[13].

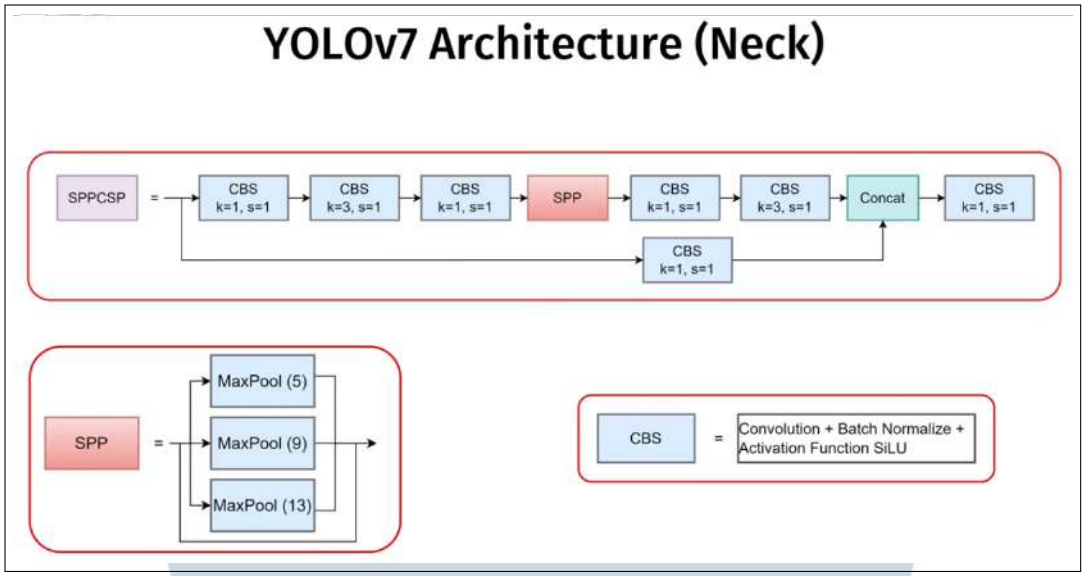
B Flowchart YOLOv7 Bagian Neck

Bagian *Neck* bertanggung jawab untuk memperkaya *enhancement* fitur-fitur yang sudah diekstrak oleh *Backbone* sebelum diteruskan ke bagian *Head*. Bagian *Neck* terdiri dari beberapa bagian *bottom-up* dan beberapa bagian *top-down*.



Gambar 2.2. *Flowchart Neck* YOLOv7

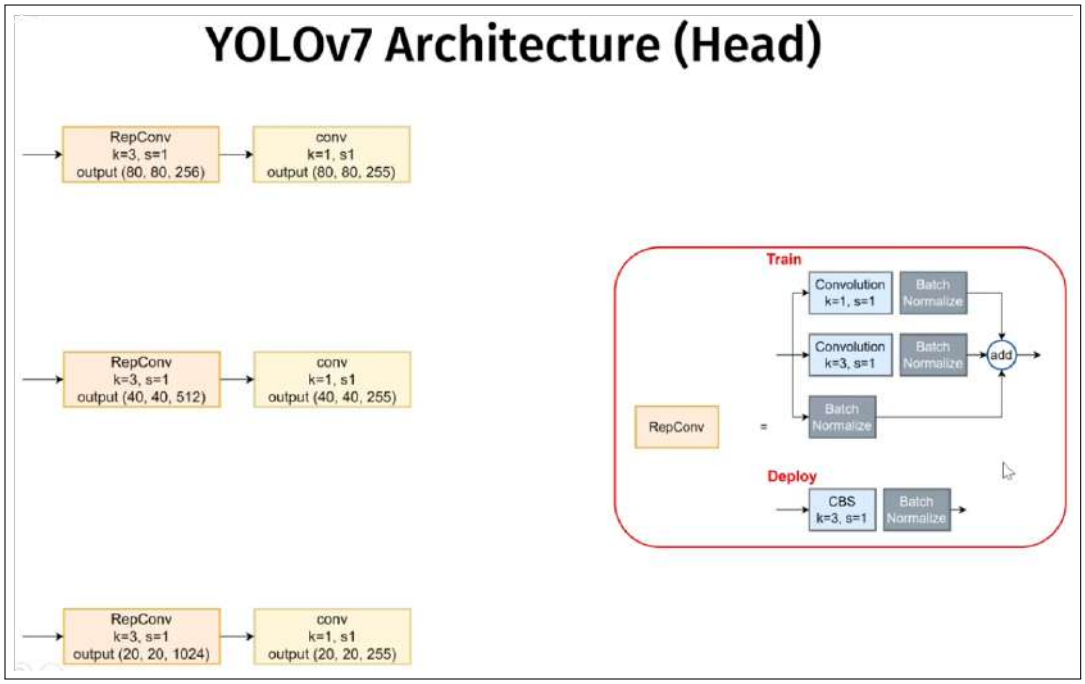
Pada Gambar 2.2 bagian *Neck* YOLOv7 memiliki struktur yang disebut dengan *Path Aggregation Feature Pyramid Network* (PAFPN). PAFPN dipilih karena PAFPN memiliki kemampuan yang baik dalam menyimpan informasi spasial yang membantu dalam proses lokalisasi piksel. Bagian *Neck* YOLOv7 memiliki sebuah blok SPPCSP yang merupakan kumpulan dari beberapa blok CBS dan sebuah blok SPP. Tujuan dari blok SPPCSP adalah untuk memperkaya informasi dari fitur-fitur yang didapat dari *Backbone*. Selain blok SPPCSP, bagian *Neck* YOLOv7 memeiliki beberapa blok ELAN dan *upsample*. Gambar 2.3 merupakan gambaran arsitektur dari blok SPPCSP.



Gambar 2.3. Arsitektur blok SPPCSP

C Flowchart YOLOv7 Bagian Head

Pada bagian *Head* dilakukan prediksi untuk lokalisasi dan klasifikasi objek dengan cara mengaplikasikan lapisan konvolusional pada fitur-fitur yang dikirim dari bagian *Neck*.

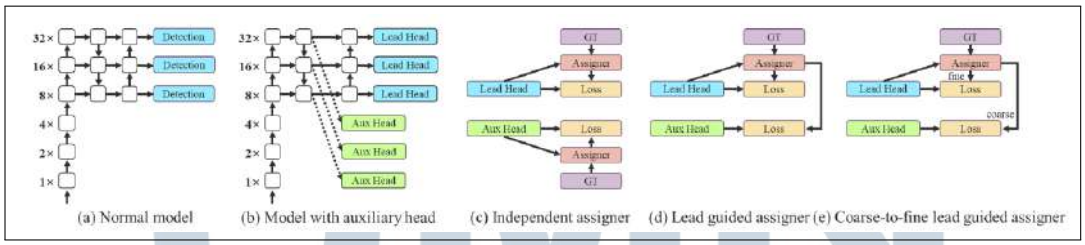


Gambar 2.4. *Flowchart Head* YOLOv7

Berdasarkan Gambar 2.4 bagian *Head* YOLOv7 mengintegrasikan 3 skala berdasarkan data dari *Neck* dan mengalokasikan 3 *anchor box* pada setiap skala. Penggunaan skala pada YOLOv7 bertujuan untuk meningkatkan akurasi pada saat mendeteksi ukuran objek yang berbeda (kecil, sedang, besar). Kemudian, pada bagian *Head* YOLOv7 terdapat sebuah lapisan RepConv yang digunakan untuk mengubah nilai dari *output channel* yang dikirim dari *Neck*. RepConv memiliki konfigurasi yang berbeda pada saat pelatihan dan inferensi. Pada saat pelatihan terdapat proses penambahan(aditif) dari ketiga cabang sedangkan pada saat inferensi, parameter dari cabang akan di *re-parameterized* kedalam sebuah cabang utama

2.3.2 Coarse for Auxiliary dan Fine for Lead Loss

Pada arsitektur YOLOv7, bagian *head* yang berfungsi untuk memberikan *final output* disebut dengan *lead head* dan bagian *head* yang berfungsi untuk membantu dalam proses *training* disebut dengan *auxiliary head*. Pada YOLOv7, hasil prediksi dari *lead head* digunakan sebagai arahan untuk membuat *coarse-to-fine hierarchical labels*. Label tersebut digunakan oleh *lead head* dan *auxiliary head* untuk belajar pada saat *training*.



Gambar 2.5. *Label Assigner* YOLOv7

Berdasarkan Gambar 2.5, peneliti YOLOv7 menggunakan 2 metode baru untuk melakukan *assignment* label. Pertama *Lead head guided label assigner*, dengan membiarkan *auxiliary head* untuk langsung mempelajari informasi yang sudah dipelajari *lead head*, *lead head* dapat berfokus untuk informasi residu yang masih belum dipelajari. Kedua *Coarse-to-fine lead head guided label assigner*, merupakan mekanisme yang memungkinan untuk *importance* dari *fine label* dan *coarse label* untuk diubah secara dinamis selama proses *learning* dan secara konsisten membuat batas atas dari *fine label* lebih tinggi daripada *coarse label*[13].

2.3.3 YOLOv7-tiny

YOLOv7 memiliki beberapa varian model, salah satunya adalah YOLOv7-tiny. Model YOLOv7-tiny dikhususkan untuk digunakan pada *edge GPU*. Varian *tiny* dari model YOLOv7 mengindikasikan bahwa varian model tersebut disesuaikan atau dioptimalkan untuk *edge AI* dan lebih ringan untuk menjalankan pembelajaran mesin pada perangkat komputasi *mobile* dan perangkat *edge* terdistribusi. Untuk pengaplikasian model di dunia nyata, varian *tiny* menjadi penting karena tidak memerlukan spesifikasi perangkat keras sebesar varian non-*tiny*. Karena YOLOv7-tiny adalah model yang berorientasi untuk digunakan pada *edge GPU*, YOLOv7-tiny menggunakan *leaky ReLu* sebagai fungsi aktivasinya. Sedangkan varian model YOLOv7 yang lain menggunakan fungsi aktivasi SiLU [13].

2.4 Confusion Matrix

Confusion Matrix adalah sebuah konsep *machine learning* yang memuat informasi tentang nilai aktual dan hasil dari klasifikasi yang dilakukan oleh model klasifikasi. Informasi yang terdapat didalam *Confusion Matrix* dapat membantu untuk menyempurnakan klasifikasi atau perkiraan yang diturunkan dari klasifikasi tersebut. *Confusion Matrix* mempunyai dua dimensi, di mana indeks pertama adalah indeks dari kelas aktual sebuah objek dan indeks kedua adalah indeks dari kelas hasil prediksi model klasifikasi. Performa dari model klasifikasi yang dihasilkan dievaluasi dengan menggunakan data yang ada pada matriks tersebut [14]. Representasi dari sebuah *Confusion Matrix* dapat dilihat pada Tabel 2.1.

Tabel 2.1. Confusion Matrix

Aktual	Prediksi	
	Negatif	Positif
Negatif	TN	FP
Positif	FN	TP

Berdasarkan Tabel 2.1, *True Negative* (TN) adalah nilai untuk prediksi benar bahwa sebuah objek bernilai negatif, *False Negative* (FN) adalah nilai untuk prediksi salah bahwa sebuah objek bernilai negatif, *False Positive* (FP) adalah nilai untuk prediksi salah bahwa sebuah objek bernilai positif, dan *True*

Positive (TP) adalah nilai untuk prediksi benar bahwa sebuah objek bernilai positif. Dengan menggunakan TN, FN, FP, TP, beberapa metriks seperti *precision, recall, specificity, accuracy,* dan *F1 score* dapat dihitung [15].

2.4.1 Precision

Precision atau presisi adalah perbandingan dari nilai untuk prediksi benar bahwa sebuah objek bernilai positif dengan keseluruhan prediksi sebuah objek bernilai positif. Nilai presisi diperoleh melalui persamaan pada Rumus 2.1 berikut.

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

2.4.2 Recall

Recall atau *Sensitivity* adalah proporsi prediksi benar dari objek yang bernilai positif dengan keseluruhan prediksi benar. *Recall* dapat diartikan sebagai kemampuan sebuah model klasifikasi untuk memilih kelas tertentu dari kumpulan data [14]. Nilai *Recall* diperoleh melalui persamaan pada Rumus 2.2 berikut.

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

2.5 IoU (Intersection over union)

IoU atau *Intersection over union* merupakan nilai dari area yang tumpang tindih antara 2 *bounding box*. IoU seringkali digunakan untuk mengukur seberapa besar area tumpang tindih antara *bounding box* hasil prediksi dengan *bounding box* yang sebenarnya. Untuk mendapatkan nilai IoU, diperlukan nilai dari area yang tumpang tindih antara 2 *bounding box* atau *intersection* dan nilai total dari area yang tercakup oleh kedua *bounding box* termasuk area yang tumpang tindih atau *Union*. Setelah mendapatkan nilai *Intersection* dan *Union*, nilai IoU dapat dihitung dengan Rumus 2.3 berikut [16].

$$IoU = \frac{Intersection}{Union} \tag{2.3}$$

2.6 mAP (mean average precision)

mAP atau *mean average precision* sebuah nilai pengukuran yang mengevaluasi nilai dari *precision* dan *recall* dari sebuah model deteksi objek. Pada sebuah model deteksi objek, nilai mAP yang semakin tinggi mengindikasikan performa deteksi yang baik. Nilai dari mAP didapatkan dari nilai *Average Precision* dari setiap kelas berdasarkan kurva *precision-recall*, kemudian di rata-rata dengan nilai AP dari seluruh kelas. Nilai dari AP didapatkan melalui Rumus 2.4 berikut. Dimana nilai R_n dan P_n merupakan nilai dari *recall* dan *precision* pada batas n [17].

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{2.4}$$

Pada model YOLOv7 digunakan 2 buah variasi mAP, yaitu mAP@0.5 dan mAP@0.5:0.95. mAP@0.5 menghitung nilai dari mAP menggunakan batas IoU sebesar 0.5. Apabila nilai IoU dari hasil prediksi dengan *ground truth* bernilai lebih besar atau sama dengan 0.5 maka hasil prediksi tersebut dinyatakan benar. mAP@0.5:0.95 menghitung nilai rata-rata dari mAP dalam batas IoU dari 0.5 hingga 0.95 dengan *step size* sebesar 0.05. Untuk setiap batas IoU, AP akan dihitung dan kemudian nilai AP setiap kelas akan di rata-rata untuk mendapatkan mAP pada batas IoU tersebut. mAP@0.5:0.95 dapat memberikan evaluasi yang lebih komprehensif untuk mengetahui preforma model dalam batas IoU yang berbeda-beda [18].

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA