BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Sebelumnya

Penelitian yang berkaitan dengan deteksi alat pelindung diri ini, bukanlah hal yang baru. Namun sudah banyak penelitian serupa yang telah dilakukan, akan tetapi dalam penelitian sebelumnya menggunakan dataset, metode, dan fungsi optimasi yang berbeda-beda. Oleh karena itu dengan memahami peneltian sebelumnya dapat membantu penulis dalam menambah refrensi untuk mengembangkan penelitian terkait dan menghindari adanya suatu duplikasi penelitian. Sehingga penelitian ini dapat berkontribusi dalam bidang ilmu pengetahuan.

Penelitian pertama berjudul "*Analysis of Object Detection Models on Duckitown Robot Based on YOLOv5 Architectures*" sebagai acuan yang dilakukan oleh Toan-Khoa Nguyen, Lien T. Vu, Viet Q. Vu, Tien-Dat Hoang Shu-Hao Liang, Member, RST, and Minh-Quang Tran*, Member, RST rilis pada 4 Desember 2021. Penelitian ini menggunakan sebuah robot yang diberi nama "Duckietown" dan memiliki peran untuk mendeteksi sebuah objek pada sebuah video, dimana objek yang dideteksi terdapat 2 class yaitu class cones dan class duckies. Robot Duckietown dibekali perangkat hardware meliputi papan onboard Raspberry Pi dan Nvidia Jetson Nano, lensa kamera depan fish-eye, 2 DC motors, 32 GB memory card, dan baterai 5V. Pelatihan model deteksi menggunakan jumlah dataset sebanyak 1000 sample dengan pembagian 80% untuk data train dan 20% untuk data validasi. Model yang dilatih dalam penelitian tersebut menggunakan beberapa variasi ukuran YOLOv5 seperti tipe ukuran small, medium, large, extra large dan fungsi optimasinya menggunakan optimasi SGD. Kesimpulan dari penelitian tersebut menunjukan bahwa arsitektur YOLOv5l (*large*) dengan fungsi optimasi SGD memiliki tingkar akurasi terbaik sebesar 97.8% (Nguyen et al., 2021)

Penelitian kedua berjudul "*Detection of mold on the food surface using YOLOv5*" dilakukan oleh Fahad Jubayer, Janibul Alam Soeb, Abu Naser Mojumder, Mitun Kanti Paul, Pranta Barua, Shahidullah Kayshar, Syeda Sabrina

Akter, Mizanur Rahman, Amirul Islam rilis pada 24 Juli 2021. Pada penelitian tersebut peneliti menggunakan model untuk mendeteksi objek jamur pada makanan khususnya jamur pada buah tomat dan roti, dimana model objek deteksi yang dilatih berupa YOLOv5s dengan dataset sebesar 850 gambar yang didapat dari laboratory of Department of Food Engineering and Technology of Sylhet Agricultural University, Sylhet, Banglades dan 1200 gambar dari sumber internet baik berbayar maupun gratis, dan hyperparameter yang digunakan meliputi image size 640, batch size 10, dan epochs 205. Kesimpulan dari penelitian tersebut menjelaskan bahwa objek yang diteliti memiliki kesulitan yang cukup tinggi dikarenakan warna objek jamur seringkali mendekati warna dari objek buah tomat dan roti sehingga dalam deteksi sering terjadi kesalahan. Oleh karena itu saran dari penulis untuk memberikan tambahan jumlah dataset untuk mendapatkan hasil akurasi yang lebih baik (Jubayer et al., 2021).

Penelitian ketiga berjudul "*Deep Learning-based Face Mask Detection Using YoloV5*" dilakukan oleh Jirarat Ieamsaard, Surapon Nathanael Charoensook, Suchart Yammen rilis pada tahun 2021. Dalam penelitian ini penulis berusaha untuk mengaplikasikan arsitektur YOLOv5 untuk mendeteksi penggunaan masker, dimana model dilatih untuk mengenali 3 class yaitu penggunaan masker yang benar, penggunaan masker yang salah, dan yang tidak menggunakan masker. Penulis juga melakukan percobaan dengan model yang memiliki variasi epoch yang berbeda-beda, dimana skenario epoch yang digunakan meliputi 20, 50, 100, 300, 500. Kesimpulan dalam penelitian tersebut menyatakan bahwa semakin tinggi epoch yang digunakan maka semakin tinggi akurasinya akan tetapi dalam kasus ini epoch 300 memili tingkat akurasi yang lebih baik dari epoch 500, dimana epoch 300 mendapat tingkat akurasi sebesar 96,5% (Ieamsaard et al., 2021).

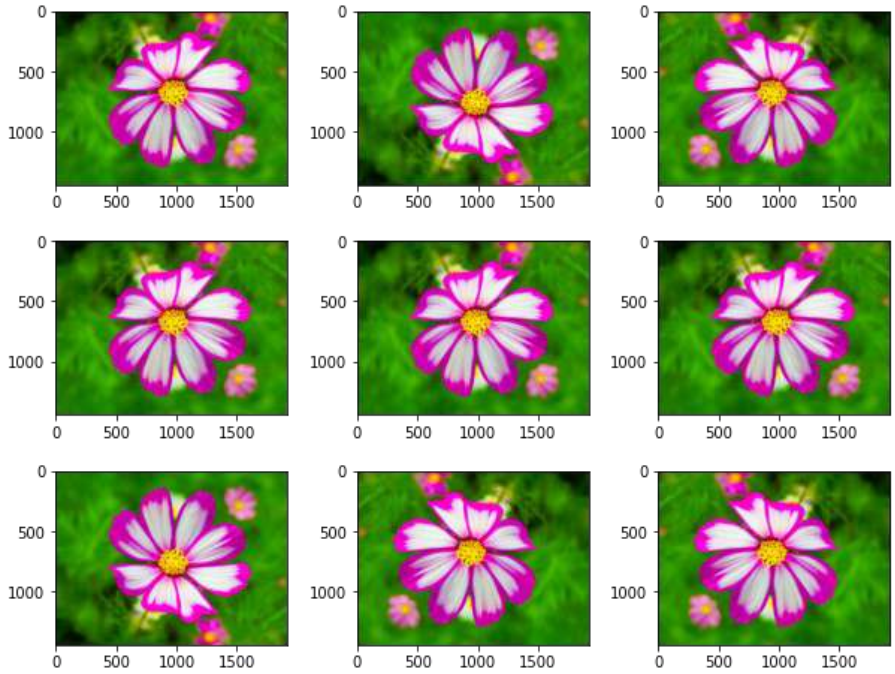
Penelitian Keempat berjudul "*Malaria Disease Detection Using CNN Technique with SGD, RMSprop, and ADAM*" dilakukan oleh Avinash Kumar, Sobhangi Sarkar and Chittaranjan Pradhan rilis pada tahun 2020. Pada penelitian tersebut peneliti menggunakan algoritma CNN untuk mendeteksi penyakit malaria yang diklasifikasikan menjadi 4 class meliput P. falciparum, P. ovale, P. vivax and P. Malariae. Penggunaan algoritma CNN dalam penelitian ini dilakukan

dalam 3 fungsi optimasi yang berbeda antara lain fungsi optimasi SGD, RMSprop, dan ADAM. Dalam proses training model peneliti menggunakan data sebanyak 27.558 gambar. Kesimpulan dari penelitian tersebut didapatkan komparasi bahwa fungsi optimasi ADAM mendapatkan hasil terbaik dengan nilai 96,88% dan selanjutnya fungsi optimasi SGD mendapatkan akurasi sebesar 95.33% dan terakhir fungsi optimasi RMSprop sebesar 95.32% (Kumar et al., 2020).

Berdasarkan referensi penelitian sebelumnya yang telah dibahas, maka penulis akan menjadikannya sebagai acuan dalam mengerjakan penelitian ini. Tujuan dari penelitian ini adalah untuk mengetahui perbandingan fungsi optimasi pada arsitektur YOLOv5 dalam mengenali objek citra alat pelindung diri berdasarkan kelas yang telah ditentukan.

2.2 Augmentasi

Augmentasi adalah sebuah proses memodifikasi atau memanipulasi citra, dimana citra asli dalam bentuk standar akan diubah bentuk dan posisinya. Augmentasi juga memiliki manfaat untuk meningkatkan keragaman data yang ada untuk proses pelatihan, tanpa benar-benar harus menambah data baru. Hal ini sangat berguna untuk mengatasi permasalahan train model yaitu data hungry (membutuhkan banyak data). Augmentasi data memiliki banyak jenis tekniknya seperti flipping, cropping, cutoff, copy-paste, padding, dll. Oleh karena banyaknya jenis teknik augmentasi maka data baru akan lebih banyak didapatkan dengan mengkombinasikan teknik augmentasi yang ada. Sehingga dalam satu citra bisa memiliki banyak jenis augmentasinya yang menyebabkan proses pembelajaran mesin dapat belajar mengenali berbagai citra yang berbeda beda sehingga bisa meningkatkan performa dari suatu model (Sanjaya & Ayub, 2020).



Gambar 2.1 Contoh Hasil Augmentasi (Setiawan, Adi., 2021)

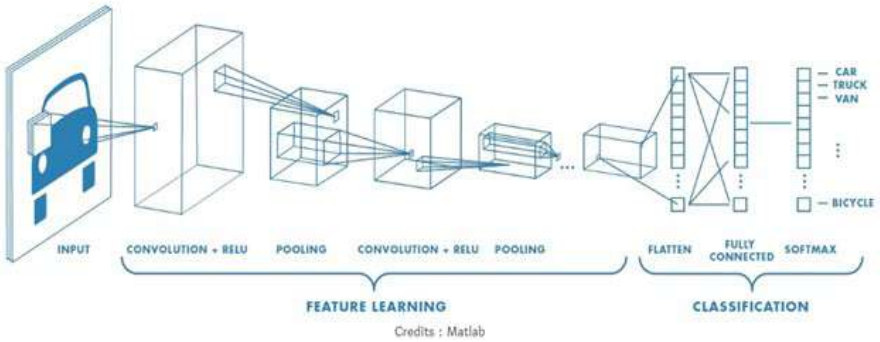
Gambar 2.1 adalah contoh sederhana dari Augmentasi Flip. Flip merupakan salah satu teknik augmentasi yang sering digunakan, dimana fungsi flip yaitu untuk membalik sebuah citra baik secara vertikal maupun horizontal. Pada gambar citra bunga di atas dilakukan metode flip horizontal dan flip vertikal.

2.3 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah algoritma yang digunakan dalam pengenalan objek, klasifikasi citra serta pengelompokan CNN merupakan algoritma deep learning yang merupakan evolusi dari *Multi-Layer Perceptron* (MLP). Saat ini, hasil terpenting dalam pengenalan gambar disediakan oleh metode CNN. Ini karena CNN mencoba meniru sistem pengenalan gambar di korteks visual manusia sehingga mereka dapat memproses informasi gambar (Putra et al., 2021).

CNN memiliki kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Penemuan CNN pertamakali di dasari oleh penelitian yang dilakukan oleh Hubel dan Wiesel (Hubel & Wiesel, 1968). Selanjutnya CNN dikembangkan pertama kali oleh Kunihiko Fukushima oleh seorang peneliti dari NHK Broadcasting Science Research Laboratories, Tokyo, Jepang dengan nama NeoCognitron (Fukushima, 1980). Konsep tersebut lalu populerkan oleh LeChun

yang merupakan seorang peneliti dari AT&T Bell Laboratories di Holmdel, New Jersey, USA dengan nama LeNet (LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, 1986). Secara teknis, CNN merupakan arsitektur yang dapat dilatih dan memiliki beberapa tahapan seperti input (masukan) dan output (keluaran) dimana setiap tahapnya memiliki beberapa array yang secara umum disebut feature map. Setiap tahap ini terdiri dari tiga lapisan meliputi lapisan konvolusi, fungsi aktivasi, dan lapisan pooling.



Gambar 2.2 Arsitektur CNN (Das, Angel., 2020)

2.4 *You Only Look Once* (YOLO)

Algoritma pemgenalan object yang utama saat ini iadalah seri R-CNN dan seri YOLO. *Region Convolutional Neural Network* (R-CNN) menggunakan *Region Proposal Network* (RPN) dimana cara kerjanya dengan membuat proposal bounding box terlebih dahulu, kemudian menjalankan pengklasifikasian pada bounding box dan selanjutnya menghapus deteksi duplikat dan memperbaiki bounding box. Karena alur kerja ini yang dilakukan satu persatu sebanyak proposal yang membuat RCNN sulit dan lambat untuk di optimalkan. Sedangkan YOLO merupakan algoritma dengan pendekatan baru dalam *object detection / computer vision* menggunakan jaringan syaraf tiruan (CNN). YOLOv1 diperkenalkan pertama kali melalui jurnal yang berjudul "You Only Look Once: Unified, Real-Time Object Detection" oleh Joseph Redmon, SantoshDivvala, Ross Girshick, dan Ali Farhadi pada tahun 2015 (Redmon et al., 2016). Ide pertama dari kreator YOLO adalah dengan mendesign sebuah algoritma yang memanfaatkan jaringan tunggal *Convolutional Neural Network* yang berguna memprediksi beberapa bounding box dan probabilitas kelas untuk bounding box tersebut, dan YOLO berlatih pada citra penuh secara langsung mengoptimalkan

kinerja pendeteksian, yang dimana dengan cara ini bisa mengalahkan kecepatan algortima pendahulunya seperti R-CNN, Fast R-CNN, Faster R-CNN.

Tabel 2.1 Perbandingan Kecepatan Model *Object Detection*

Model	Kecepatan	
R-CNN	0.05 fps	20 s / gambar
Fast R-CNN	0.5 fps	2 s / gambar
Faster R-CNN	7 fps	140 ms / gambar
YOLO	45 fps	22 ms / gambar

Kerangka kerja YOLO dibagi menjadi 3 bagian utama, yaitu :

- Backbone : Convolutional Neural Network yang menggabungkan dan membentuk fitur gambar pada berbagai jenis detail gambar.
- Neck : Serangkaian lapisan jaringan yang mencampur dan menggabungkan fitur gambar dan meneruskannya ke lapisan prediksi.
- Head : Menggunakan fitur dari Neck untuk menghasilkan kotak pembatas, dan memprediksi kategori.

2.4.1 Cara Kerja YOLO

Pertama, Algoritma YOLO akan membagi citra dalam grid berukuran SxS, apabila dalam region S terdapat objek maka region tersebut akan melakukan deteksi. Setiap region akan memprediksi bounding box dan peta kelas pada masing masing grid (Justitian et al., 2022). Sehingga setiap region akan memiliki bounding box dan nilai confidence, secara umum nilai confidence memiliki persamaan sebagai berikut :

$$Confidence = Pr(Object) * IOU \quad (2.1)$$

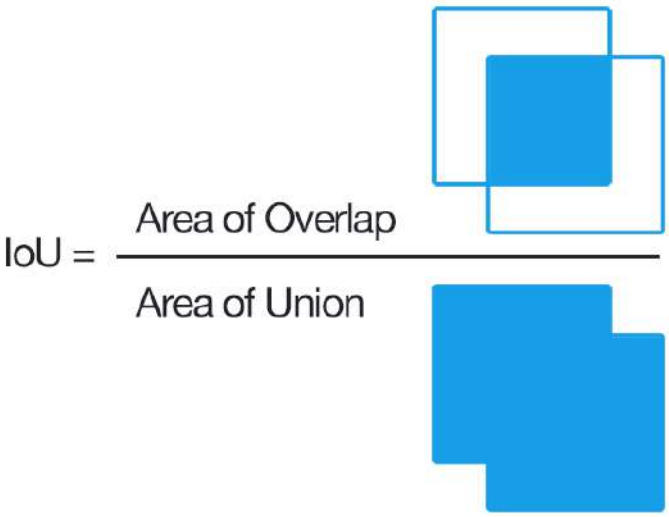
(Putra et al., 2021) Pr($Object$) merupakan nilai objektivitas, apabila tidak terdapat objek maka akan berinilai nol dan bernilai satu jika terdapat sebuah objek. *IOU* merupakan nilai rasio *Intersection Over Union* antara kotak *ground*

truth dan kotak prediksi. Truth merupakan area kepastian objek sedangkan pred merupakan area kotak prediksi.

Setiap Bounding Box berisi 5 nilai prediksi yaitu *x, y, w, h, conf*. Nilai *x, y* mewakili pusat kotak terhadap batas batas dari region. Nilai *w, h* mewakili nilai lebar dan tinggi dari keseluruhan citra. Sedangkan *conf* mewakili nilai confidence antara kotak ground truth dan kotak prediksi.

Selain itu grid cell juga memprediksi probabilitas kelas *Pr(Classi |Object).* Probabilitas diberikan pada region yang terdapat sebuah objek, satu probabilitas hanya akan diberikan per satu region grid, terlepas dari jumlah kotak prediksi. Pada saat pengujian, YOLO mengkalkulasi probabilitas kelas kondisional dengan prediksi kotak confidence dengan persamaan:

$$Pr(Class\ i\ |Object)* Pr(Object)*IOU= Pr(Class\ i)*IOU \quad (2.2)$$



Gambar 2.3 *Intersection Over Union* (Hofesmann, Eric., 2020)

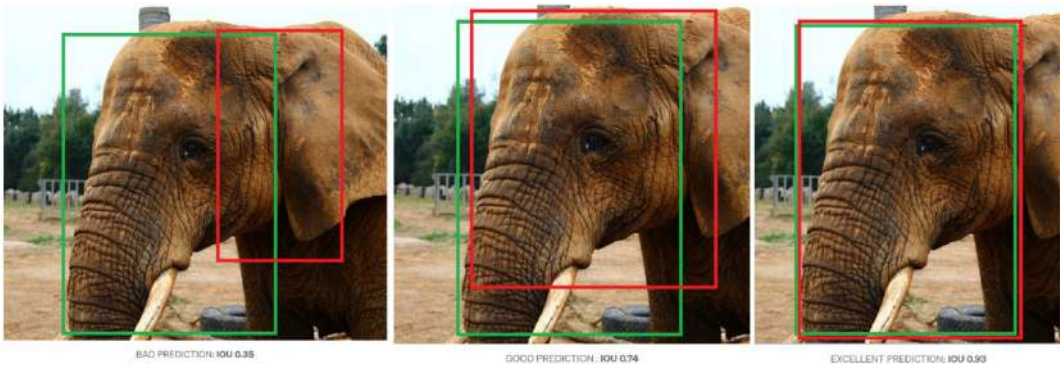
IoU atau *Intersection Over Union* merupakan metrik yang mengukur akurasi model dalam mendeteksi objek pada kumpulan data latih. IoU membandingkan bounding box ground truth dan bounding box prediksi yang terletak pada suatu objek dalam citra gambar. Nilai IoU yang dihasilkan akan berfungsi sebagai nilai confidence untuk ke tahap proses *Non Max Supression*. IoU dapat didefinisikan dengan persamaat berikut:

$$\text{IoU} = \frac{A \cap B}{A \cup B} \tag{2.3}$$

Keterangan :

A = Bounding Box Ground Truth

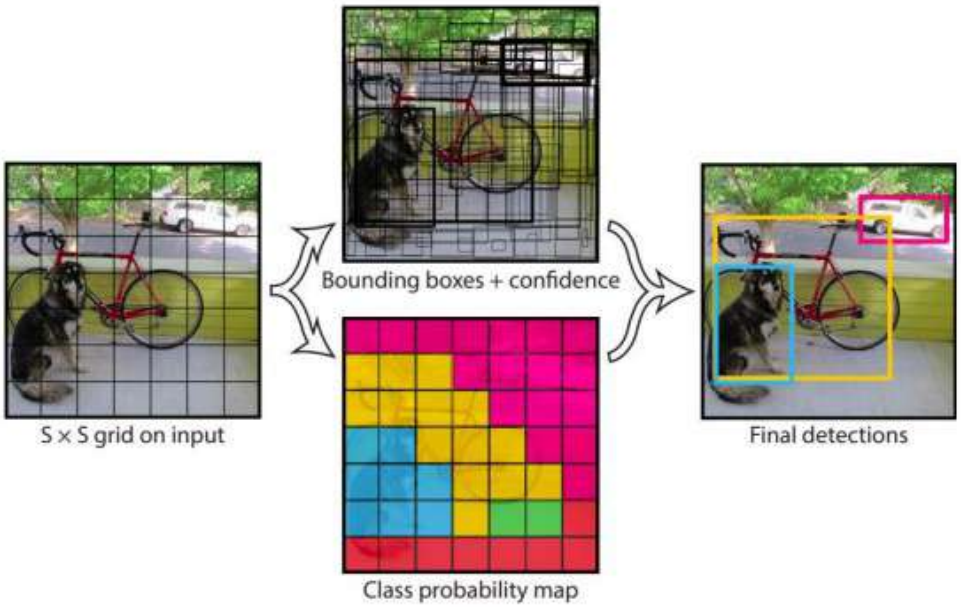
B = Bounding Box Prediksi



Gambar 2.4 Contoh Hasil IOU

Gambar 2.4 adalah contoh dari IOU. IOU memiliki rentang nilai antara 0 dan 1. Nilai IOU yang mendekati 0 dianggap sebagai prediksi yang buruk, sedangkan nilai IOU yang mendekati 1 dianggap sebagai prediksi yang baik. Umumnya nilai thresshold secara default akan di set 0.5, akan tetapi dalam penerapannya nilai thresshold bisa di set berapapun sesuai kebutuhan penelitian. Biasanya nilai IOU>5 akan dianggap sebagai prediksi yang baik. Dalam gambar diatas, gambar pertama dianggap sebagai bad prediction karena nilai IOU<0.5, gambar kedua dianggap sebagai good prediction karena nilai IOU>0.5, sedangkan gambar ketiga dianggap sebagai excellent prediction karena nilai IOU mendekati 1 atau IOU>0.9.

Non Max Supression berfungsi untuk menghilangkan bounding box yang overlap / tumpang tindih terhadap objek dalam suatu gambar. Non Max Supression akan memanfaatkan dua nilai yaitu IoU dan Threshold. Threshold merupakan nilai yang ditentukan oleh peneliti sebagai ambang batas. Apabila Nilai IoU dari suatu bounding box lebih kecil dari nilai Threshold maka bounding box tersebuh akan di hilangkan (Arwindo et al., 2020).



Gambar 2.5 YOLO Conceptual Design (Redmon et al., 2016).

Loss function / cost function merupakan fungsi yang digunakan untuk mengukur seberapa baik performa yang dihasilkan model dalam mendeteksi sebuah objek. Fungsi loss yang baik adalah fungsi yang menghasilkan error yang paling rendah. Jika model memiliki kelas lebih dari dua / multiclass maka perlu adanya cara untuk mengukur perbedaan nilai antara probabiliras hasil prediksi dan probabilitas nilai aktual.

Fungsi loss yang digunakan dalam algoritma YOLO adalan Mean Square Error (MSE). Mean Square Error (MSE) merupakan fungsi loss yang paling sering umum untuk digunakan dan sering disebut juga sebagai L2 Loss, dimana cara kerjanya yaitu dengan menghitung rata rata perbedaan kuadrat nilai aktual dan nilai prediksi. Hasil dari MSE akan selalu bernilai positif terlepas dari nilai prediksi dan nilai aktualnya. Untuk mencapai model dengan performa terbaik, peneliti harus mencoba mengurangi nilai L2 Loss sekecil mungkin atau mencapa nilai sempurna yaitu 0 (Arwindo et al., 2020). Berikut merupakan rumus persamaan untuk menghitung nilai MSE :

$$\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \epsilon classes} \left(P_i(c) - \left(\hat{P}\right)_i(c)\right)^2 \qquad (2.4)$$

Keterangan :

1. S = jumlah grid
2. 1_i^{obj} = Objek, bernilai 1 jika terdapat objek, dan bernilai 0 jika tidak terdapat objek
3. $P_i(c)$ = Nilai prediksi
4. $(\hat{P})_i(C)$ = Nilai aktual

2.4.2 Deskripsi YOLOv5

Tidak lama setelah YOLOv4 rilis pada April 2020, Gleen Jocher dan tim Ultralytics LLC melakukan publikasi versi terbaru dari Seri YOLO, versi terbaru tersebut diberi nama YOLOv5 (Jocher, 2020). Gleen Jocher merupakan peneliti, pengembang dan CEO Ultralytics LLC.

Secara umum arsitektur YOLOv5 tidak jauh berbeda dengan YOLO versi sebelumnya. YOLOv5 dikembangkan menggunakan bahasa pemrograman Pyhton, bukan seperti YOLO versi sebelumnya yang menggunakan bahasa pemrograman C. Hal ini yang membuat instalasi dan integrasi pada perangkat IoT lebih mudah digunakan. Selain itu YOLOv5 di kembangkan menggunakan kerangka library PyTorch dimana memiliki komunitas yang lebih besar dari komunitas Darknet, yang menyebabkan PyTorch akan mendapatkan dukungan kontribusi dan potensi pertumbuhan yang lebih baik di masa depan (Thuan, 2021).

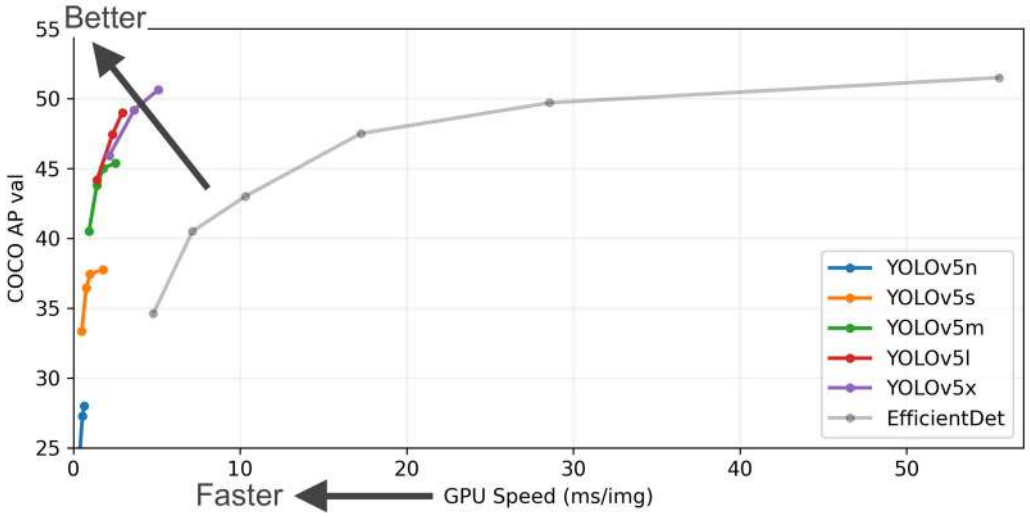
Meskipun dikembangkan dalam dua bahasa pemrograman dan kerangka kerja yang berbeda, kinerja antara YOLOv4 dan YOLOv5 tidak jauh berbeda, akan tetapi setelah beberapa saat YOLOv5 terbukti lebih berperforma baik akurasi maupun kecepatan daripada YOLOv4 dalam beberapa kasus, dan juga sebagian komunitas computer vision lebih percaya akan performa YOLOv5 (Thuan, 2021).

YOLO memiliki tiga kerangka kerja utama yaitu *backbone, neck* dan *head.* Bagian *backbone* YOLOv5 menggunakan CSPDarknet53 yang di adopsi dari CSPDarknet YOLOv4, bagian ini memecahkan pengulangan informasi gradien di *backbone* dan mengintegrasikan perubahan gradien ke dalam peta fitur sehingga dapat meningkatkan akurasi, mengurangi kecepatan *inferensi*, dan mengurangi ukuran bobot model dengan mengurangi jumlah parameter. Bagian *neck* YOLOv5

menggunakan *Path Aggregation Network* (PANet) untuk meningkatkan arus informasi. PANet ini pengembangan dari Feature Pyramid Network (FPN) di YOLOv3 yang mencakup lapisan bottom up dan top down, dengan ini dapat meningkatkan propagasi level rendah fitur dalam model. PANet meningkatkan lokalisasi di lapisan bawah, yang meningkatkan akurasi lokalisasi objek. Bagian head YOLOv5 sama dengan YOLOv3 dan YOLOv4 yang menghasilkan tiga keluaran peta fitur yang berbeda untuk mencapai skala multi prediction, sehingga dapat membantu dalam mengingkatkan prediksi objek kecil hingga besar secara efisien (Nepal & Eslamiat, 2022).

2.4.3 Tipe YOLOv5

YOLOv5 memiliki beberapa tipe antara lain YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, dan masing-masing tipe tersebut memiliki perbedaan dari segi kecepatan deteksi dan performa akruasi mAPnya. Pada penelitian ini Penulis akan menggunakan YOLOv5. Berikut gambar dari masing-masing tipe YOLOv5.

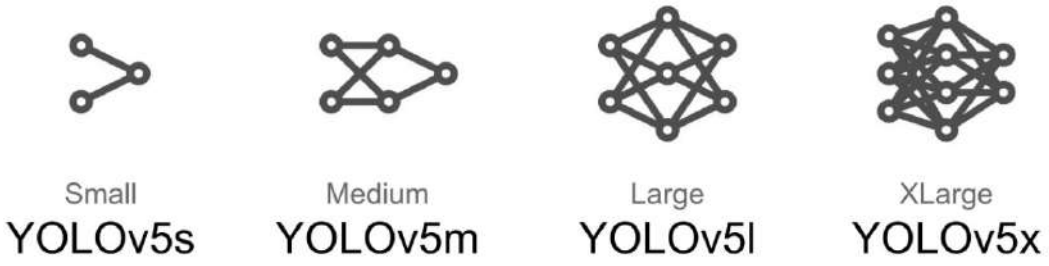


Gambar 2.6 Tipe YOLOv5 (Glenn Jocher, 2022)

Tabel 2.2 Perbandingan Performa YOLOv5

Model	Size (Pixels)	mAP 0.5:0.95	mAP 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	Params (M)	FLOPS @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

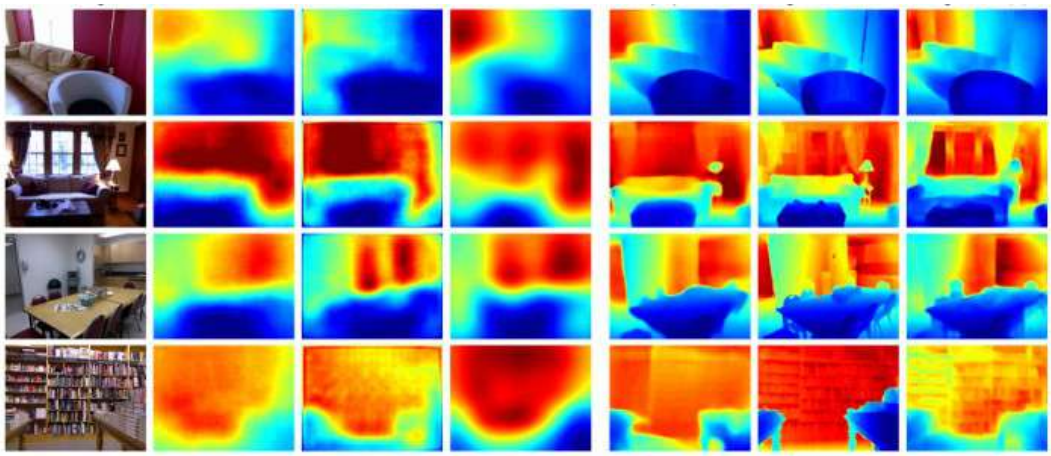
Pada Gambar 2.6 menunjukan bahwa semakin ke kiri arah grafik maka semakin cepat deteksi, dan semakin keatas grafik maka akan semakin baik akurasi mAP model. Pada tabel 2.2 menunjukan beberapa indikator yang bisa menjadi bahan pertimbangan dalam memilih model. Model YOLOv5 dilatih dengan dataset COCO dengan size 640 piksel. Semakin besar ukuran tipe YOLOv5 maka akan semakin besar juga nilai akurasi mAP, kecepatan CPU, dan params. Semakin tinggi tipe YOLO maka kecepatan CPUnya semakin cepat akan tetapi tipe YOLO yang tinggi memiliki jumlah params yang jauh lebih besar, sehingga membuat komputasi akan lebih lama dan proses deteksi akan semakin lambat.



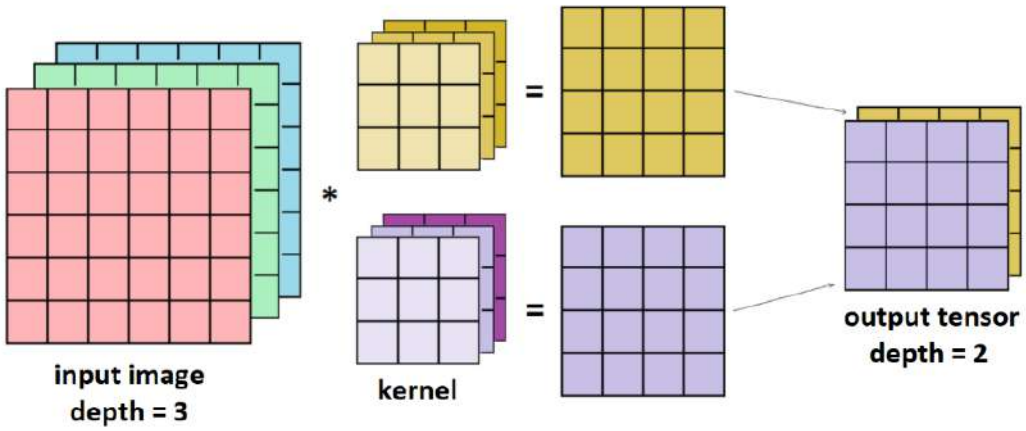
Gambar 2.7 Parameter Tipe YOLO (Glenn Jocher, 2022)

Pada gambar 2.7 merupakan gambaran parameter dari tipe YOLO. Parameter merupakan variabel konfigurasi internal model yang nilainya diberikan dari data pretrained model yang digunakan. Parameter ini digunakan untuk mementukan keterampilan model dalam mebuat prediksi (Glenn Jocher, 2022). Semakin tinggi tipe YOLO maka akan memiliki jaringan neural network yang lebih banyak sehinga parameter yang digunakan semakin besar.

Perbedaan tipe YOLO tersebut didasarkan pada jumlah depth multiple dan layer channel multiple (Tan M et al, 2020). Umumnya input neural network merupakan citra berwarna. Gambar tersebut terdiri dari tiga saluran yang mewakili insensitas warna merah, hijau, dan biru. Setiap piksel dalam citra menggabungkan ketiga warna tersebut, di mana intensitas warna digambarkan dengan bilangan bulat dari 0 hingga 255. Oleh karena itu, gambar input memiliki lebar, tinggi, dan kedalamannya. Kedalaman gambar input menentukan kedalaman lapisan input (Enes Zvornicanin, 2022).



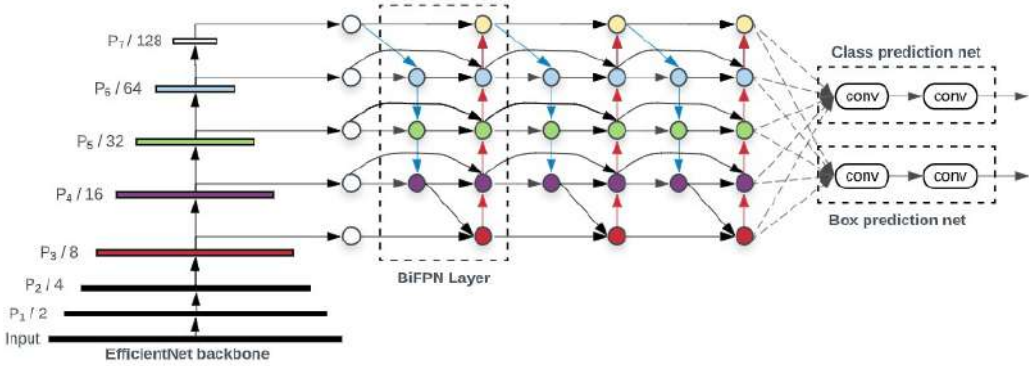
Gambar 2.8 Contoh Depth Image (Derrick Mwiti, 2019)



Gambar 2.9 Ilustrasi Depth (Enes Zvornicanin, 2022)

Selain menggunakan (*Depth*) kedalaman input gambar yang lebih besar atau menumpuk lebih banyak lapisan (*Layer Channel*). Metode ini biasanya tidak efektif karena hanya berfokus pada satu dimensi penskalaan yang terbatas. Oleh karena itu, YOLO akan melakukan pendekatan penskalaan lain (*Compound*

Scaling), tetapi tetap mengikuti gagasan utama untuk memperbesar semua dimensi secara bersama-sama (Tan M et al., 2020).



Gambar 2.10 *Compund Scaling*

Compound Scaling digunakan untuk mengoptimasi dan menefisiensikan prediksi model. Proses ini dilakukan dengan cara anchor box akan melakukan scale up aspek ratio dengan maksud untuk menyesuaikan ukuran objek yang akan di deteksi, dengan contoh sebagai berikut :

a). P3/8 digunakan untuk mendeteksi objek yang lebih kecil

b). P4/16 digunakan untuk mendeteksi objek sedang

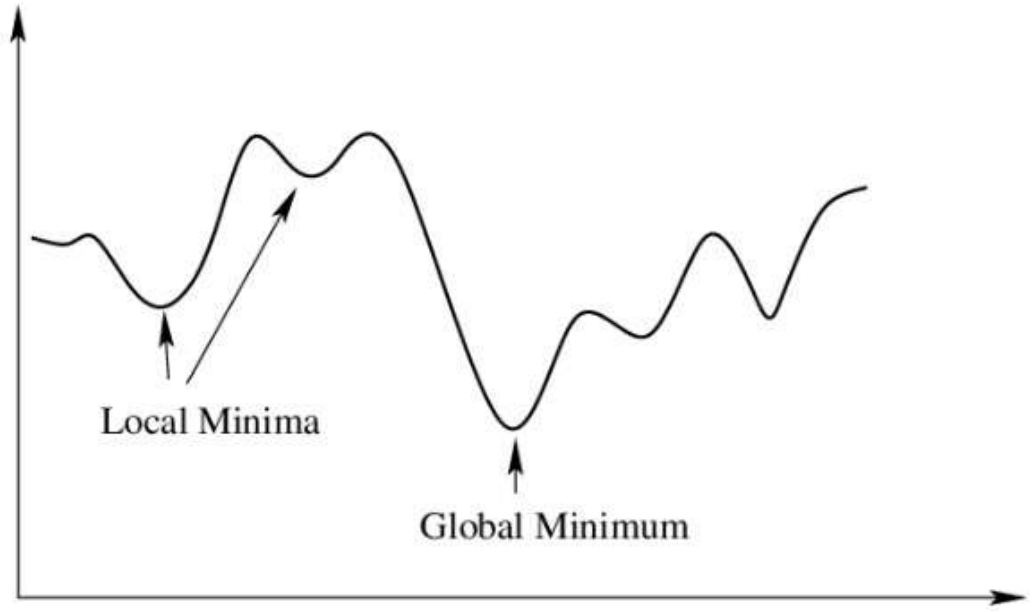
c). P5/31 digunakan untuk mendeteksi objek yang lebih besar

Oleh karena itu dalam prosesnya akan dimulai dari P1 hingga ke P7. Hal ini dilakukan untuk melakukan deteksi pada object terkecil terlebih dahulu menggunakan anchor box yang lebih kecil, selanjutnya akan diteruskan dengan ukuran anchor box sedang hingga ke paling besar.

2.5 Optimizer

Pada deep learning model belajar dengan cara memperbarui parameter dengan cara membuat nilai loss sekecil mungkin. Cara yang digunakan untuk membantu model belajar dengan adanya penggunaan optimizer. Optimizer berfungsi untuk membantu meningkatkan akurasi dan meminimalkan nilai loss / cost function. Titik dimana loss function mengambil nilai minimun disebut sebagai global minimum. Namun pada proses penggunaan *optimizer* sering kali ditemukan tampak nilai minimum pada titik yang berbeda, akan tetapi nilai tersebut tidak mengambil nilai minimum loss function, maka titik tersebut disebut sebagai local minima (Kumar Ajitesh, 2021). Nilai global minimum yang baik

merupakan nilai yang konvergen ke 0 (mendekati nilai 0) dan pada beberapa iterasi terakhirnya nilainya tidak terupdate atau tidak mengalami update yang besar.



Gambar 2.11 Ilustrasi *loss function* (Kumar Ajitesh, 2021)

2.5.1 SGD Optimizer

SGD atau Stochastic Gradient Descent biasa merupakan algoritma optimasi yang mencari suatu bobot baru dengan metode pengambilan salah satu data dari semua data training, sesudah itu SGD melakukan analisa dari setiap data yang diambil. Memakai SGD dapat mengurangi penggunaan memori yang diperlukan disaat pemrosesan bobot baru dan dalam sehingga proses pembelajaran model lebih cepat, akan tetapi dalam proses pembelajaran model akan sering terjadi fluktuasi yang cukup tinggi (Ruder, 2016). Berikut adalah proses langkah – langkah algoritma SGD sebagai berikut :

1. Menambah t pada setiap iterasi

$$t = t + 1 \tag{2.5}$$

2. Menghitung nilai gradien

$$g_t = \frac{\delta L}{\delta W} \tag{2.6}$$

3. Menghitung weight average

$$v_t = \gamma . v_{t-1} + \eta . g_t \tag{2.7}$$

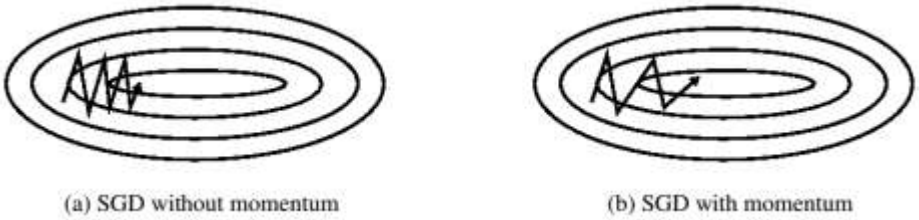
4. Memperbarui parameter

$$\theta = \theta_{t-1} - v_t \tag{2.8}$$

Keterangan :

1. t = iterasi
2. g = gradien
3. δL= turunan fungsi loss
4. δW= turunan weight
5. v = weight average
6. γ= momentum
7. η= learning rate
8. θ= parameter yang di perbaiki

Momentum adalah metode yang dapat membantu mempercepat SGD ke arah yang relevan dan meredam osilasi seperti yang terdapat pada Gambar 2.5. Ini dilakukan dengan menambahkan parameter momentum γ dari vektor pembaruan step terakhir ke vektor pembaruan saat ini (Ruder, 2016). Nilai momentum untuk penelitian ini di set 0.937.



Gambar 2.12 Momentum

2.5.2 Adam Optimizer

Adam atau Adaptive Moment Estimation merupakan algoritma pengembangan dari Adagrad dimana cara kerjanya mirip kombinasi antara RMSProp dan Momentum, disebut adaptive karena mirip RMSProp dimana learning rate berubah sepanjang training, dan moment estimation karena menggunakan estimasi momen pada matematika. Adam pertama kali di perkenalkan pada makalah ICLR pada tahun 2015 oleh Diederik Kingma dari University of Amsterdam, OpenAI dan Jimmy Ba dari University of Toronto dengan judul "Adam: A Method For Stochastic Optimization" (Kingma & Ba, 2017). Algoritma optimasi ini memiliki kelebihan dimana sangat efisien secara komputasi dan dalam prosesnya membutuhkan sedikit memori. Berikut adalah proses langkah – langkah algoritma Adam sebagai berikut :

1. Menambah t pada setiap iterasi

$$t = t + 1 \tag{2.9}$$

2. Menghitung nilai gradien

$$g_t = \frac{\delta L}{\delta W} \tag{2.10}$$

3. Memperbarui bias pada momen pertama

$$m_t = \beta_1 . m_{t-1} + (1 - \beta_1) . g_t \tag{2.11}$$

4. Memperbarui bias pada momen kedua

$$v_t = \beta_2 . v_{t-1} + (1 - \beta_2) . (g_t)^2 \tag{2.12}$$

5. Memperbarui koreksi bias pada momen pertama

$$\widehat{m}_t = \frac{mt}{1 - \beta_1^t} \tag{2.13}$$

6. Memperbarui koreksi bias pada momen kedua

$$\widehat{v}_t = \frac{vt}{1 - \beta_2^t} \tag{2.14}$$

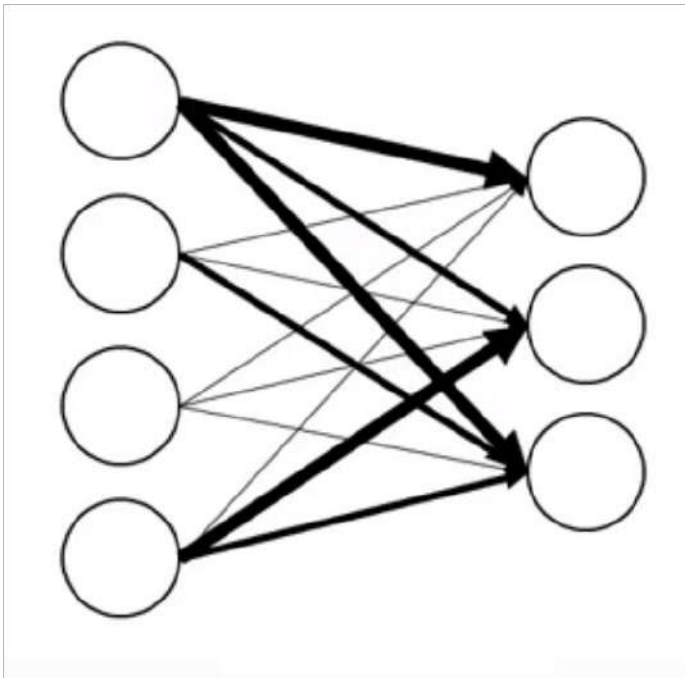
7. Memperbarui parameter

$$\theta_t = \frac{\theta_{t-1} - \alpha . \widehat{m}_t}{\sqrt{\widehat{v}_t + \varepsilon}} \tag{2.15}$$

Keterangan :

1. t = iterasi
2. δL= loss function
3. δW= bobot
4. m = momen pertama
5. v = momen kedua
6. β_1, β_2= exponential decay rates
7. g = gradien
8. θ= parameter yang diperbaiki tertinggi yaitu masing-masing
9. α= stepsize
10. ε= epsilon

Penulis menentukan nilai default untuk inisialisasi parameter Adam Optimizer 0,937 untuk β1, 0,999 untuk β2, dan 10−8 untuk ε. Mereka menunjukkan secara empiris bahwa Adam bekerja dengan baik dalam praktik dan lebih baik dibandingkan dengan algoritma metode pembelajaran adaptif lainnya (Ruder, 2016).



Gambar 2.13 Learning Rate berubah sepanjang proses training

Pada gambar 2.13 merupakan gambaran learning rate yang berubah pada proses training. Oleh sebab learning rate yang bisa berubah sepanjang proses training ini menyebabkan model akan lebih lebih cepat untuk belajar, karena data yang sering dilatih / diupdate maka learning rate akan semakin kecil yang bisa mencegah terjadinya osilasi dan untuk data yang jarang diupdate maka learning ratenya akan tetap besar sehingga untuk mencapai konvergen lebih cepat.

2.6 Confusion Matrix

Confusion Matrix merupakan ringkasan tabel yang berfungsi untuk menganalisis performa atau kinerja dari model klasifikasi atau prediksi pada machine learning (Pratiwi et al., 2020), di dalam confusion matrix terdapat beberapa nilai seperti akurasi, recall, precision, dan f1 score dari suatu model (Ghoneim, 2019). Tabel confusion matrix ditunjukkan sebagai berikut :

Tabel 2.3 Confusion Matrix

		Ground Truth	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

Keterangan Tabel :

1. True Positif (TP) merupakan jumlah data yang bernilai positif, diprediksi dengan nilai benar oleh model.
2. True Negatif (TN) merupakan jumlah data yang bernilai negatif, namundiprediksi positif oleh model.
3. False Negatif (FN) merupakan jumlah data yang bernilai negatif, yangdiprediksi negatif oleh sistem.
4. False Positif (FP) merupakan jumlah data yang bernilai positif, namundiprediksi negatif oleh model.

2.6.1 Akurasi

Akurasi merupakan rasio dari jumlah data yang diprediksi sesuai kelasnya dari keseluruhan data. Nilai akurasi dapat diperoleh menggunakan persamaan berikut ini:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

2.6.2 Presisi

Presisi adalah rasio dari jumlah data yang sesuai kelasnya dibandingkan dengan data yang diprediksi positif. Nilai presisi dapat diperoleh menggunakan persamaan berikut ini :

$$Precision = \frac{TP}{TP + FP} \tag{2.17}$$

2.6.3 Recall

Recall adalah rasio dari jumlah data yang diprediksi sesuai kelasnya dibandingkan dengan data yang memang sesuai kelasnya. Nilai recall dapat diperoleh menggunakan persamaan berikut ini :

$$Recall = \frac{TP}{TP + FN} \tag{2.18}$$

2.6.4 F1 Score

F1 score adalah rasio perbandingan rata-rata presisi dan recall. Nilai F1 Score dapat diperoleh menggunakan persamaan berikut ini :

$$F1 = \frac{2\,X\,(RecallXPrecision)}{(Recall + Precision)} \tag{2.19}$$

2.7 Evaluasi MAP

Evaluasi digunakan untuk mengukur seberapa baik sebuah model dalam melakukan deteksi objek. Mean Average Precision (mAP) adalah sebuah metrik yang digunakan untuk mengukur tingkat akurasi deteksi objek suatu model (Henderson et al., 2016). Mean Average Precision (mAP) juga merupakan metrik populer yang sering digunakan oleh algoritma lain seperti Faster R-CNN, SSD, YOLO, dll. Average Precision(AP) ini akan menghitung nilai average precision untuk nilai penarikan antara 0 hingga 1. Karena pada penelitian ini menggunakan *multi class classification* maka average precision akan dihitung dengan menggunakan binary classifier untuk setiap kelasnya. Kemudian nilai dari setiap kelasnya dirata-rata sehingga mendapat angka mAPnya. Berikut merupakan rumus persamaan untuk menghitung Mean Average Precision (mAP) :

$$AP = \frac{1}{11} \sum_{r \epsilon \{0,0,..,1,0\}} APr \tag{2.20}$$

$$AP = \frac{1}{11} \sum_{r \epsilon \{0,0,...,1,0\}} p_{interp}(r) \tag{2.21}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} APi \tag{2.22}$$

Keterangan :

1. N = Jumlah Data AP
2. AP = Average Precision
3. p = presisi
4. r = recall