BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Penelitian yang pernah dilakukan sebelumnya penting untuk membantu melakukan penelitian tentang deteksi objek pada tanaman hias Miana. Beberapa penelitian memiliki keterkaitan dengan penelitian yang sedang penulis lakukan dan sebagai bahan referensi penulis.

Tabel 2.1 Penelitian Terkait

No	Judul Penelitian	Peneliti	Hasil	Perbedaan
1	Implementasi Algoritma Yolo (You Only Look Once) Untuk Deteksi Api	(Nazilly et al., 2020)	Pada penelitian tersebut YOLO mampu mendeteksi objek dengan baik dengan nilai *confidence* 0.66 dan terendah 0.55 dengan nilai tertinggi 0.71.	Pada penelitian tersebut peneliti hanya mendeteksi satu jenis objek yaitu api.
2	Deteksi Jenis Mobil Menggunakan Metode YOLO Dan Faster R-CNN	(Shianto et al., 2019)	Pada penelitian tersebut algoritma YOLO memiliki tingkat akurasi yang lebih rendah namun prediksi yang tepat dari Faster R-CNN.	Pada penelitian tersebut selain menggunakan 2 metode yaitu YOLO dan Faster R-CNN kemudian menggabungkan dua metode tersebut untuk mendeteksi.

3	Pengenalan Objek Makanan Cepat Saji Pada Video Dan Real Time Webcam Menggunakan Metode *You Look Only Once* (YOLO)	(Karlina & Indarti, 2019)	Hasil deteksi pada makanan cepat saji pada penelitian tersebut dengan menggunakan video dan secara *real-time* memperoleh akurasi 63% sampai 100% dengan 3 jenis makanan yang berbeda.	Pada penelitian yang penulis lakukan menggunakan objek daun tanaman Miana.
4	Deteksi Sampah pada Real-time Video Menggunakan Metode Faster R- CNN	(Rahman & Bambang, 2021)	Pada penelitian tersebut menghasilkan tingkat keberhasilan deteksi objek sampah sebesar 74% dengan metode Faster R-CNN.	Pada penelitian penulis menggunakan metode *You Only Look Once* (YOLO).
5	Deteksi Tanaman Tebu Pada Lahan Pertanian Menggunakan Metode Convolutional Neural Network	(Muhammad Alfin Jimly Asshiddiqie, Basuki Rahmat, 2020)	Pada penelitian tersebut deteksi tanaman tebu menghasilkan akurasi sebesar 0.95 dengan nilai *precision* 1.0 dan *recall* 0.95.	Pada penelitian tersebut deteksi dilakukan pada video yang diambil menggunakan drone kemudian dideteksi untuk membedakan lahan yang ditanami tebu, sedangkan pada

		penelitian yang penulis lakukan deteksi objek dilakukan secara *real-time.*

Penelitian yang dilakukan oleh penulis memiliki perbedaan terletak pada objek penelitian yaitu tanaman miana dengan jumlah 10 jenis. Penerapan algoritma YOLO menggunakan YOLOv4 *tiny* yaitu versi terbaru dari YOLO yang memiliki kecepatan tinggi dalam proses pendeteksian. Selain itu pada penelitian ini deteksi objek dilakukan dengan perangkat android untuk deteksi objek secara *real time*.

2.2 Tanaman Miana

Tanaman Miana (*Coleus*) adalah salah satu tanaman tumbuhan semak yang dapat ditemukan di daerah tropis dan subtropis, terdapat banyak jenis dari tanaman *Coleus* yang ditemukan. Pada penelitian (Paton et al., 2019) tanaman *Coleus* memiliki 294 jenis. *Coleus* ditemukan oleh João de Loureiro pada tahun 1970. Tanaman Miana sangat mudah untuk dibudidayakan yaitu dengan cara generatif maupun dengan cara vegetatif. Selain memiliki daun yang unik dan indah warnanya, Miana populer karena memiliki harga yang terjangkau. Miana memiliki variasi motif dan warna yang berbeda-beda setiap jenisnya. Tanaman Miana sering kali dimanfaatkan untuk hiasan rumah dan teras oleh para pecinta tanaman hias.

2.3 Pengolahan Citra Digital

Definisi citra adalah gambaran duplikasi dari sebuah objek nyata yang berupa *output* suatu sistem perekaman data yang dapat disimpan dalam sebuah memori penyimpanan. Terdapat dua jenis citra yaitu citra analog dan citra digital. Citra analog adalah citra yang memiliki sifat kontinu seperti gambar monitor tv, foto sinar X, lukisan, *CT scan* yang tidak dapat diproses langsung oleh komputer. Sedangkan citra digital adalah citra yang dapat diolah dengan komputer. Citra digital memiliki nilai-nilai yang disimpan

pada komputer berupa angka-angka yang menunjukan intensitas dari piksel pada citra tersebut.(Sutoyo et al., 2009). Pada citra digital terdapat sistem pencitraan atau *imaging* yaitu proses mengubah citra analog menjadi citra digital, proses tersebut disebut dengan proses digitalisasi citra. Digitalisasi citra bertujuan mengubah citra analog agar bisa menjadi citra digital yang dapat diproses menggunakan komputer. Alat yang dapat digunakan untuk mengubah citra dari analog ke digital adalah kamera digital, kamera konvensional dan scanner.

2.4 Machine Learning

Machine learning adalah bagian dari kecerdasan buatan yang bertujuan agar komputer mampu belajar layaknya manusia. *Machine learning* diciptakan agar membantu manusia untuk menganalisa sebuah data untuk menghasilkan sebuah prediksi. *Machine learning* melakukan training data untuk pembelajaran kemudian menghasilkan pola-pola dan ciri kemudian melakukan klasifikasi dan melakukan pengujian (Huang et al., 2006). Beberapa teknik dalam *machine learning* adalah sebagai berikut:

1. *Supervised learning* merupakan pembelajaran mesin yang berdasarkan data yang diberikan sehingga mendapatkan sebuah kesimpulan dari pembelajaran berdasarkan dataset yang sudah ada.
2. *Unsupervised learning* merupakan pembelajaran mesin yang tidak menggunakan data latih sebagai acuan.
3. *Reinforcement learning* adalah pembelajaran mesin yang mampu belajar secara otomatis dari aksi-aksi yang dilakukan oleh sebuah agent.

2.5 Deep Learning

Deep learning merupakan cabang dari kecerdasan buatan dan machine learning. Deep learning merupakan *artificial neural network* yang memanfaatkan layer-layer untuk pengolahan informasi masukan non-linear untuk kebutuhan pengenalan pola, ekstraksi fitur pada citra, dan proses klasifikasi kelas (Deng & Yu, 2014). *Deep learning* dimanfaatkan pada

pengembangan machine learning seperti pada studi pengenalan objek, pengenalan suara dan pengenalan sebuah visual (LeCun et al., 2015).

Deep learning merupakan cara untuk menirukan pemikiran manusia yang memanfaatkan *artificial neural network*. *Deep learning* juga merupakan arsitektur dari *supervised learning.*

2.6 Computer Vision

Computer vision adalah salah satu cabang keilmuan komputer dengan tujuan untuk memberikan kemampuan melihat suatu objek. *Contoh penerapan computer vision* adalah *face recognition, object detection, dan color detection. Computer Vision* adalah salah satu kemampuan komputer untuk mempelajari sebuah citra berupa gambar atau video, komputer akan menganalisa *input* citra yang diberikan kemudian komputer memberikan informasi yang didapatkan dari *input* yang diberikan. *Computer vision* diciptakan memiliki tujuan untuk mengikuti kemampuan dari mata manusia menafsirkan visual yang dilihat (Purno & Wibowo, 2016). (Boyle & Thomas, 1988) memberikan pengertian pada *Computer Vision* dengan sebuah *image recognition* yang menggunakan operasi algoritma low level processing dan mengelompokan pengolahan citra sebagai *Computer Vision.* Menurut Ballard dan Brown (1982) *Computer vision* adalah proses integrasi pemrosesan dari representasi suatu visual dengan melalui beberapa tahap tertentu.

2.7 Algoritma You Only Look Once (YOLO)

Algoritma *You Only Look Once* (YOLO) merupakan salah satu algoritma yang digunakan untuk pengenalan objek. Algoritma ini diciptakan oleh (Redmon et al., 2016). Algoritma ini diciptakan untuk memperbaiki algoritma deteksi objek sebelumnya. YOLO memanfaatkan *Convolution Neural Network* untuk mendeteksi objek. Proses pendeteksian dengan YOLO memiliki 3 langkah yaitu:

1. Mengubah ukuran citra input.
2. Menjalankan *single convolutional network* pada input citra.

3. Menerapkan *threshold* berdasarkan nilai *confidence*.

YOLO membagi input citra menjadi grid-grid dengan ukuran S x S. Pada setiap grid bertanggung jawab untuk memprediksi adanya objek dan *bounding box* dan *confidence*. Nilai *confidence* akan memprediksikan bahwa adanya sebuah objek. Setiap *bounding box* terdapat parameter yaitu (x, y, w, h, dan *confidence*). Parameter x dan y adalah koordinat dari bounding box. Parameter w dan h adalah tinggi dan lebar *bounding box*. *Confidence* adalah nilai dari *Intersection over Union* dari perhitungan dari *predicted box* dan *ground-truth box*. Pada grid akan memprediksi probabilitas kelas.

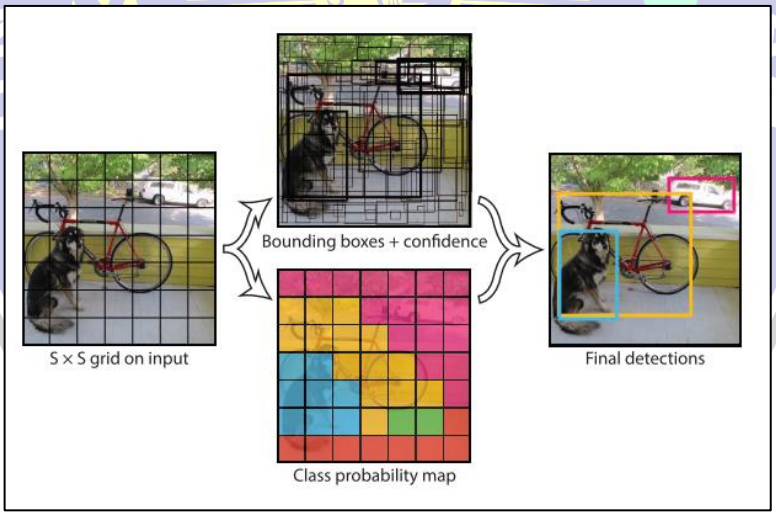
$$Pr\ (Class_i|Object)\ x\ Pr\ (Object)\ x\ IoU_{pred}^{truth} = Pr\ (Class_i)\ x\ IoU_{pred}^{truth} \quad (2.1)$$

Keterangan:

$Pr\ (Class_i|Object)$ = probababilitas kondisional kelas i.

$Pr\ (Object)$ = probababilitas kelas i.

IoU_{pred}^{truth} = *intercetion over union*.

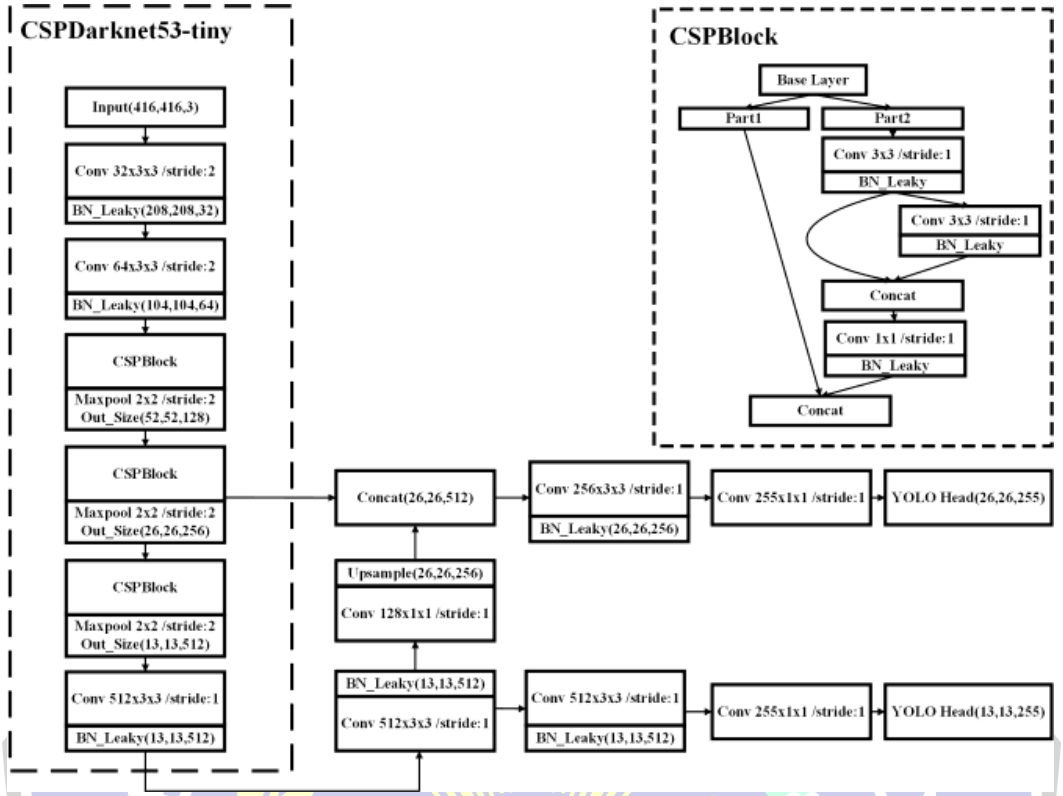


Gambar 2.1 Deteksi YOLO

Sumber: You Only Look Once: Unified, Real-Time Object Detection (Redmon et al., 2016)

Metode YOLO memiliki kecepatan deteksi yang mampu mengungguli metode deteksi objek lainnya. Dalam penelitian yang dilakukan (Redmon & Farhadi, 2017) YOLO mampu memiliki mengungguli dalam kecepatan algoritma deteksi objek lain Faster R-CNN, ResNet, dan SSD. Pada Yolov2 kecepatan 67 fps dan mendapatkan mAP sebesar 76,8 pada VOC 2007. Pada 40 fps YOLO mampu mendapatkan mAP sebesar 78,6 mengungguli Faster R-CNN, ResNet dan SSD.

Metode *You Only Look Once* (YOLO) yang dirancang oleh (Redmon et al., 2016) dengan versi terakhir darinya yaitu YOLOv3 yang menggunakan framework darknet-53 dan kemudian berhenti untuk memperbarui YOLOv3 dan dilanjutkan oleh (Bochkovskiy et al., 2020) untuk pengembangan terbaru yaitu YOLOv4. Pengembangan YOLOv4 telah disetujui oleh pengembang YOLOv3. Dalam pengembangan YOLOv4 mampu mengungguli YOLOv3 dengan memperoleh AP dan FPS sebesar 10% dan 12% lebih tinggi. Kecepatan deteksi YOLOv4 meningkat drastis jika dibandingkan dengan YOLOv3.



Gambar 2.2 Struktur Jaringan YOLOv4-tiny

Sumber: Real-time object detection method for embedded devices (Jiang et al., 2020)

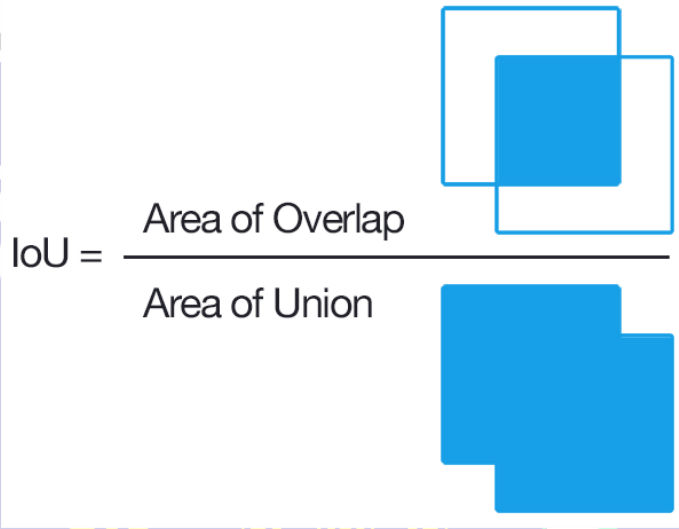
Struktur jaringan YOLOv4-tiny pada Gambar 2.2 adalah CSPDarknet53-tiny atau versi kecil dari CSPDarknet53 yang digunakan pada YOLOv4. CSPDarknet53-tiny menggunakan modul CSPBlock terbagi menjadi dua feature map dan kemudian menggabungkannya. Modul CSPBlock dapat meningkatkan kemampuan belajar jaringan konvolusi dan meningkatkan komputasi yang menyebabkan meningkatknya akurasi metode YOLOv4-tiny. Penggunaan dua feature map yang berbeda yaitu 13x13 dan 26x26 untuk memprediksi hasil deteksi (Jiang et al., 2020).

Pada penelitian ini penulis menggunakan YOLOv4-tiny untuk perancangan aplikasi deteksi tanaman miana. YOLOv4-tiny dirancang berdasarkan YOLOv4 untuk meningkatkan kecepatan deteksi. Deteksi dengan YOLOv4-tiny mampu mendeteksi objek dengan kecepatan 371 fps menggunakan GPU 1050Ti dengan akurasi yang sangat baik. YOLOv4-tiny

sangat layak digunakan untuk pendeteksian pada perangkat kecil atau seluler. Prediksi yang dilakukan oleh YOLOv4-tiny sama dengan YOLOv4 (Jiang et al., 2020).

Intersection over Union (IoU) digunakan untuk menghitung tingkat presisi bounding box. Dibutuhkan 2 *bounding box* untuk perhitungan IoU. Perhitungan *Intersection over Union* (IoU) adalah perhitungan antara *ground-truth* dan *predicted box*.

$$IoU_{pred}^{truth} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (2.1)$$



Gambar 2.3 Ilustrasi *Intersection over Union*

Sumber: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (Diakses pada 23 Juli 2021)

Loss function digunakan untuk menghitung loss pada kinerja yang dilakukan pada metode YOLO.

$$\lambda coord \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \quad (2.2)$$

$$\lambda coord \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \quad (2.3)$$

Keterangan:

$\mathbb{I}_{ij}^{obj}$ = nilai keberadaan objek

x_i, y_i = koordinat titik tengah objek yang diprediksi

$\hat{x}_i, \hat{y}_i$ = koordinat titik tengah objek aktual

w_i, h_i = lebar dan tinggi kotak pembatas

Pada persamaan 2.3 dan 2.4 digunakan untuk menghitung loss pada prediksi kotak pembatas atau *bounding box*, x dan y adalah koordinat dari prediksi objek, dan w dan h adalah lebar dan tinggi dari prediksi objek. Dimana pada persamaan 2.3 dan 2.4 $^{obj}{}_{ij}$ akan bernilai satu jika terdapat objek dan 0 apabila tidak terdapat objek.

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \left(C_i - \hat{C}_i\right)^2 \quad (2.4)$$

$$+ \lambda noobj \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} (C_i - \hat{C}_i)^2 + (C_i - \hat{C}_i)^2 \quad (2.5)$$

Keterangan:

$\mathbb{I}_{ij}^{obj}$ = nilai keberadaan objek.

$\mathbb{I}_{ij}^{noobj}$ = nilai ketidakberadaan objek.

C_i = nilai *confidence score* yang didapatkan.

$\hat{C}_i$ = nilai *confidence score* dari *intersection over union*.

Pada persamaan 2.5 dan 2.6 digunakan untuk menghitung loss confidence, Ci adalah nilai confidence score dan $\mathbb{I}^{noobj}{}_{ij}$ adalah nilai dari ketiadaan objek akan bernilai 1 dan o jika terdapat objek.

$$\sum_{i=0}^{S^2} \mathbb{I}_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \quad (2.6)$$

Keterangan:

$\mathbb{I}_{ij}^{obj}$ = nilai keberadaan objek.

$p_i(c)$ = *class* yang diprediksi.

$\hat{p}_i(c)$ = *class* aktual.

Pada persamaan 2.7 digunakan untuk menghitung loss dari klasifikasi. $\mathbb{1}^{obj}_{ij}$ akan bernilai satu jika terdapat objek dan 0 apabila tidak terdapat objek.

2.8 Android Studio

Android Studio adalah software *Intregated Development Environm*ent (IDE) untuk pengembangan aplikasi sistem operasi Android. Pada Android Studio bahasa pemrograman yang digunakan adalah Java dan Kotlin. Android Studio memiliki banyak fitur yang berguna untuk pengembangan aplikasi android. Beberapa fitur yang membantu pembuatan aplikasi Android seperti *Gradle* yang fleksibel, Emulator Android, dan lingkungan pengembangan yang menjadi satu bisa untuk semua macam Android (Suryana, 2018)

2.9 Google Colab

Google Colab *environment* dari Google yang digunakan untuk menjalankan program. Colab adalah salah satu produk dari Google yang berbasis cloud dan dijalankan melalui browser. Colab menyediakan processor dengan spesifikasi tinggi (GPU dan TPU) dengan tujuan memudahkan para researcher menjalankan program yang membutuh spesifikasi tinggi secara online (Bisong, 2019). Google menyediakan GPU *(graphics processing unit*) yang bisa digunakan secara gratis dengan tipe Nvidia K80s, P4s, T4s, dan untuk versi berbayar akan mendapatkan tipe V100 dan P100s.

2.10 Java

Java merupakan salah satu bahasa pemrograman yang diciptakan oleh James Gosling pada tahun 1991 dengan nama pertama yaitu Oak dan berganti pada tahun 1995 menjadi Java. Bahasa pemrograman Java adalah salah satu bahasa pemrograman dengan paradigma *Object Oriented Programing* (OOP). Java mampu membuat sebuah aplikasi mulai dari *mobile, desktop dan web* dan mampu dijalankan pada sistem operasi Linux, Windows, DOS, dan Unix, selain itu banyak kelebihan bahasa pemrograman Java yaitu sederhana, berorientasi objek, mudah dalam

distribusi, interpreter, aman, *architecture neutral*, *portable*, *multithreaded* dan dinamis (Emanuel Jando & Paskalis Andrianus Nani, 2018).

2.11 Python

Bahasa diciptakan oleh Guido van Rossum di belanda pada tahun 1990. Bahasa python adalah bahasa *interpreted high-level programing* yang bisa digunakan untuk berbagai tujuan. Python memiliki metode pemrosesan *interpreted* atau kode program tidak membutuhkan compile dan dijalankan baris per baris (Sembiring & Erfina, 2020)

2.12 Tensorflow

Tensorflow adalah *library open source* yang dikembangkan oleh Google Brain untuk pembelajaran mesin dan jaringan syaraf. Tensorflow mampu menjalankan algoritma-algoritma *machine learning*. Tujuan tensorflow dibuat adalah untuk riset, coding dan pengembangan pembelajaran mesin. *Library* ini dapat digunakan pada berbagai bahasa pemrograman seperti Python, Java, C/C++ (Rucci & Casile, 2015).

Tensorflow Lite dirancang oleh Google agar dapat berjalan pada perangkat kecil atau perangkat seluler. Tensorflow Lite adalah sebuah solusi bagi yang ingin menjalankan model pembelajaran mesin di perangkat seluler. Untuk menjalankan Tensorflow Lite diperlukan konversi dari model Tensorflow ke Tensorflow Lite. Mode Tensorflow Lite tersedia untuk perangkat android dan iOS

2.13 Confusion Matrix

Confusion Matrix adalah sebuah cara yang digunakan untuk mengukur kinerja algoritma yang digunakan. Pengukuran performa pada confusion *matrix* terdapat 4 paramater yang digunakan yaitu TP (*True Positive*), TN (*True Negative*), FP (*False Positive*) dan FN (*False Negative*). Dari parameter tersebut akan dihitung untuk mendapatkan metrik berupa *accuracy*, *precission* dan *recall* (Han & Kamber, 2006).

Accuracy adalah rasio hasil prediksi benar (positif dan negatif) dengan keseluruhan data.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.8)$$

Precission adalah rasio prediksi bernilai benar (positif) dengan semua data yang yang bernilai positif.

$$precission = \frac{TP}{TP + FP} \quad (2.9)$$

Recall adalah rasio dari prediksi yang bernilai benar (positif) dengan data yang memprediksi benar.

$$recall = \frac{TP}{TP + FN} \quad (2.10)$$