# BAB II

# TINJAUAN PUSTAKA

## 2.1. Landasan Teori

### 2.1.1. *Hand Gesture*

*Hand Gesture* merupakan elemen penting dalam Interaksi Manusia dan Komputer (HCI). Keyboard dan mouse merupakan perangkat input yang digunakan untuk mengoperasikan komputer, namun perangkat tersebut tidak menyediakan antarmuka yang sesuai. Sedangkan, sistem saat ini dimana *hand gesture* dapat digunakan untuk berinteraksi dengan komputer (Sharma dan Rengarajan, 2020). Banyak aplikasi dibangun dari sistem pengenalan gestur mulai dari pengenalan bahasa isyarat, kontrol robot, *virtual reality*, dan *games* (Sunyoto dan Harjoko, 2014). *Hand gesture* dapat memudahkan mobilitas dan fleksibilitas pengguna dalam menggunakan suatu perangkat atau komputer. Ditambah lagi dengan adanya pandemi Covid-19 yang mengharuskan menjaga jarak, maka pengendalian dengan sentuhan cukup memberikan rasa khawatir tersendiri. Dengan adanya pengenalan *hand gesture* dapat mengatasi kekhawatiran tersebut (Alvin dkk, 2021).

Gesture dapat dibedakan menjadi dua jenis, yaitu *static gesture* dan dynamic *gesture*. *Static gesture* merupakan posisi tangan tidak berubah selama melakukan gerakan atau postur tangan secara diam yang diwakili oleh satu gambar. Sedangkan *dynamic gesture* adalah gerakan bergerak yang diwakili oleh urutan lebih dari satu gambar (Chandra, Venu dan Srikanth, 2015).

### 2.1.2. Deteksi Objek

Deteksi objek merupakan salah satu contoh teknik di bidang *computer vision*, dimana objek dalam bentuk gambar di proses oleh algoritma *machine learning* atau *deep learning* untuk menemukan objek yang dimaksud. Deteksi objek seolah-olah mesin yang dapat melihat seperti manusia. Tujuan deteksi objek yaitu untuk meniru kecerdasan manusia pada komputer (Pardede dan Hardiansah, 2022).

Deteksi objek berbasis *deep learning* terbagi menjadi dua kategori yaitu *one-stage object detection* dan *two-stage object detection* (Zou dkk, 2023). Deteksi objek *one-stage object detection* hanya membutuhkan satu tahapan untuk melakukan proses deteksi objek. Sedangkan *two-stage object detection* mempunyai dua tahapan dalam melakukan proses deteksi objek, tahap pertama yaitu memprediksi kandidat *bounding box* pada citra dan tahap kedua dilakukan proses klasifikasi dan regresi terhadap kandidat *bounding box* yang nantinya akan menghasilkan *bounding box* yang menunjukkan letak serta tipe objek. *Two-stage object detection* meningkatkan akurasi *localization* (penentuan posisi) dan *recognition* (pengenalan tipe) objek namun memerlukan waktu yang dibutuhkan untuk melakukan proses deteksi, sebaliknya pada *one-stage object detection* mempercepat waktu proses deteksi namun mengurangi akurasi *localization* dan *recognition* (Lohia dkk, 2021).

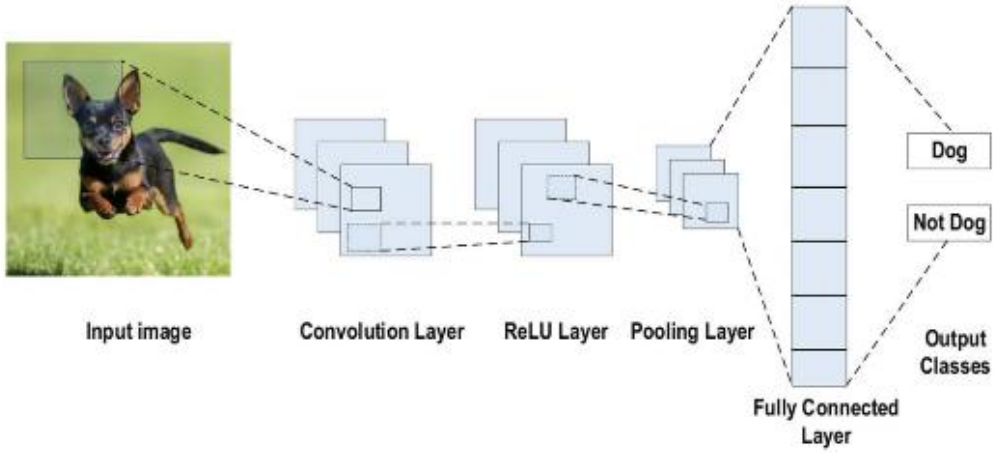
### 2.1.3. *Deep Learning*

*Deep Learning* merupakan metode pembelajaran yang terinspirasi dari otak manusia dengan menerapkan *artificial neural network* yang memiliki banyak *layer*

(Santoso dan Ariyanto, 2018). *Deep Learning* dapat digunakan dalam kasus di mana pemrosesan data yang sangat besar atau kompleks (Jamshidi dkk, 2020). *Artifical Neural Network* ini dibuat mirip otak manusia, dimana neuron-neuron terkoneksi satu sama lain sehingga membentuk sebuah jaringan neuron yang sangat rumit. *Deep Learning* dapat dipandang sebagai gabungan *Machine Learning* dengan AI (Nugroho dkk, 2020).

### 2.1.4. *Convolutional Neural Network*

*Convolutional Neural Network* (CNN) adalah evolusi dari *Multilayer Perceptrons* (MLP) yang dirancang untuk memproses data dua dimensi dalam bentuk gambar. CNN ini dikategorikan sebagai jenis *Deep Neural Network* (DNN) karena kedalaman jaringannya yang dalam dan aplikasi yang berat untuk data gambar (Nada, 2019). Secara teknis, *Convolutional Neural Network* (CNN) adalah sebuah arsitektur yang dapat dilatih dan terdiri dari beberapa tahap. Masukan dan keluaran dari setiap tahap adalah terdiri dari beberapa array yang biasa disebut *feature map*. Setiap tahap terdiri dari tiga *layer* yaitu *convolutional layer*, *activation function layer* dan *pooling layer* (Yuliany, Aradea dan Nur Rachman, 2022). Berikut ini merupakan jaringan arsitektur *Convolutional Neural Network* (CNN) yang disajikan pada Gambar 2.1.



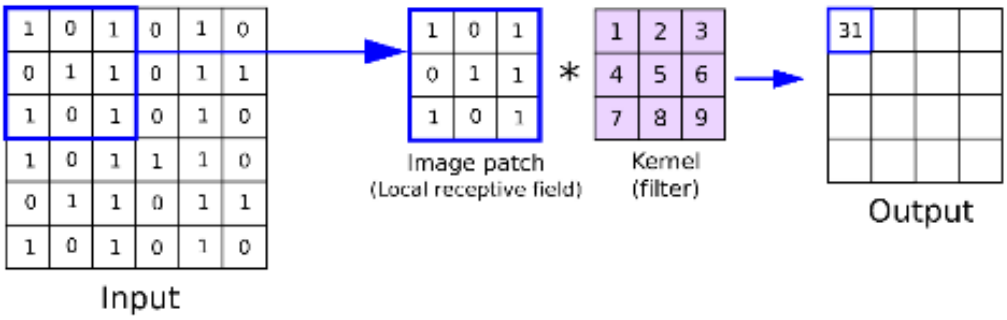

Gambar 2.1 Arsitektur CNN (Alzubaidi dkk, 2021)

Berdasarkan Gambar 2.1, Tahap pertama pada arsitektur CNN adalah tahap konvolusi. Tahap ini dilakukan dengan menggunakan sebuah kernel dengan ukuran tertentu. Perhitungan jumlah kernel yang dipakai tergantung dari jumlah fitur yang dihasilkan. Kemudian dilanjutkan menuju fungsi aktivasi, biasanya menggunakan fungsi aktivasi ReLU (*Rectifier Linear Unit*), Selanjutnya setelah keluar dari proses fungsi aktivasi kemudian melalui proses pooling. Proses ini diulang beberapa kali sampai didapatkan peta fitur yang cukup untuk dilanjutkan ke *fully connected neural network,* dan dari *fully connected network* adalah *output class* (Nada, 2019).

## A. *Convolutional Layer*

*Convolution layer* merupakan bagian dari tahap pada arsitektur CNN. Tahap ini melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari jaringan arsitektur CNN. Konvolusi adalah istilah matematis dimana pengaplikasian sebuah fungsi pada *output* fungsi lain secara berulang. Operasi konvolusi merupakan operasi pada dua fungsi argumen

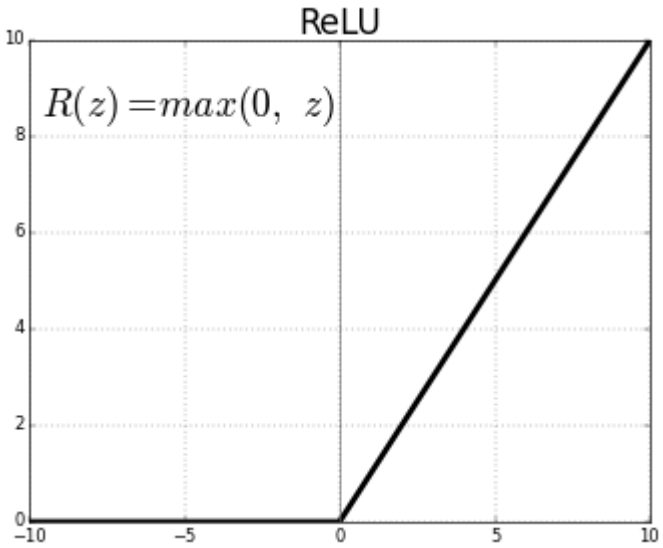
bernilai nyata. Operasi ini menerapkan fungsi *output* sebagai *feature map* dari *input* citra. *Input* dan *output* ini dapat dilihat sebagai dua argumen bernilai riil. *Convolutional Layer* terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah *filter* dengan panjang dan tinggi (*pixels*). Sebagai contoh, *layer* pertama pada *feature extraction layer* biasanya adalah conv. *Layer* dengan ukuran 5x5x3. Panjang 5 *pixels*, tinggi 5 *pixels* dan tebal/jumlah 3 buah sesuai dengan *channel* dari image tersebut. Ketiga *filter* ini akan digeser keseluruh bagian dari gambar. Setiap pergeseran akan dilakukan operasi "*dot*" antara input dan nilai dari *filter* tersebut sehingga menghasilkan sebuah *output* atau biasa disebut sebagai *activation map* atau *feature map* (Arrofiqoh dan Harintaka, 2018). Pada Gambar 2.2 merupakan ilustrasi dari *convolutional layer*.




Gambar 2.2 *Convolutional Layer* (A H. Reynolds, 2019)

B. ReLU *Activation Function Layer*

*Rectified Linear Units* (ReLU) layer ini mengaplikasikan fungsi ***f(x) = max(0,x)*** untuk meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang reseptif pada convolutional layer (Arrofiqoh dan Harintaka, 2018). Berikut ini contoh operasi ReLU layer yang diilustrasikan pada Gambar 2.3.

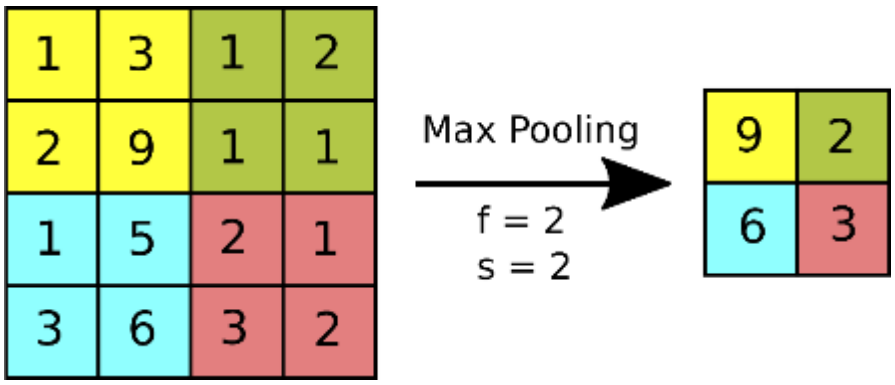



Gambar 2.3 ReLU (A H. Reynolds, 2019)

C. *Pooling Layer*

*Pooling* merupakan pengurangan ukuran matriks dengan menggunakan operasi *pooling*. *Pooling layer* biasanya berada setelah lapisan konvolusi. Pada dasarnya *pooling layer* terdiri dari sebuah *filter* dengan ukuran dan *stride* tertentu yang akan secara bergantian bergeser pada seluruh area *feature map*. Dalam *pooling layer* terdapat dua macam *pooling* yang biasa digunakan yaitu *average pooling* dan *max-pooling*. Nilai yang diambil pada *average pooling* adalah nilai rata-rata, sedangkan pada *max-pooling* adalah nilai maksimal. Lapisan Pooling yang

dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran *volume output* pada *feature map*, sehingga mengurangi jumlah parameter dan perhitungan di jaringan, untuk mengendalikan *overfitting*. Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk lapisan *pooling* umumnya dengan menggunakan *filter* dengan ukuran 2x2 yang diaplikasikan dengan langkah sebanyak dua dan beroperasi pada setiap irisan dari inputnya (Arrofiqoh dan Harintaka, 2018). Berikut ini adalah contoh gambar operasi *max-pooling* yang diilustrasikan pada Gambar 2.4.




Gambar 2.4 Max-Pooling (A H. Reynolds, 2019)

D. *Fully Connected Layer*

Lapisan *fully connected* merupakan kumpulan dari proses konvolusi. Lapisan ini mendapatkan input dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu. Fungsi dari lapisan ini adalah untuk menyatukan semua node menjadi satu dimensi (Arrofiqoh dan Harintaka, 2018).

### 2.1.5. YOLO

YOLO (*You Only Look Once*) adalah salah satu algoritma *deep learning* yang memanfaatkan *Convolutional Neural Network* (CNN) dalam mendeteksi sebuah objek dengan menggunakan *framework* bernama *darknet*. YOLO memiliki kelebihan kecepatan deteksi yang tinggi menggunakan *Convolutional Neural Network* (CNN) khusus untuk melakukan klasifikasi dan lokasi beberapa objek dalam satu gambar sekaligus (Bochkovskiy dkk, 2020).

YOLO merupakan algoritma *deep learning* untuk deteksi objek menggunakan pendekatan yang berbeda dengan algoritma lainnya yaitu menerapkan *single-layer neural network* pada keseluruhan citra. Berikut ini merupakan tahapan YOLO dalam mendeteksi objek sampai dengan memperoleh nilai *confidence*, yaitu (Nazilly dkk, 2020) :

1. Membagi citra dalam *grid* berukuran *s x s*. *Grid* tersebut bertanggung jawab untuk mendeteksi objek. Pada tiap *grid* juga akan diprediksi *bounding box* beserta nilai *confidence*. Nilai *confidence* ini menunjukkan seberapa yakin *bounding box* tersebut berisi objek dan seberapa akurat prediksinya. Nilai *confidence* diperoleh melalui persamaan berikut.

$$Conf(class) = \Pr(class) \times IoU_{Pred}^{Truth} \quad \ldots\ldots\ldots \quad (2.1)$$

Keterangan :

$\Pr(class)$ = probabilitas objek yang muncul dalam suatu region.

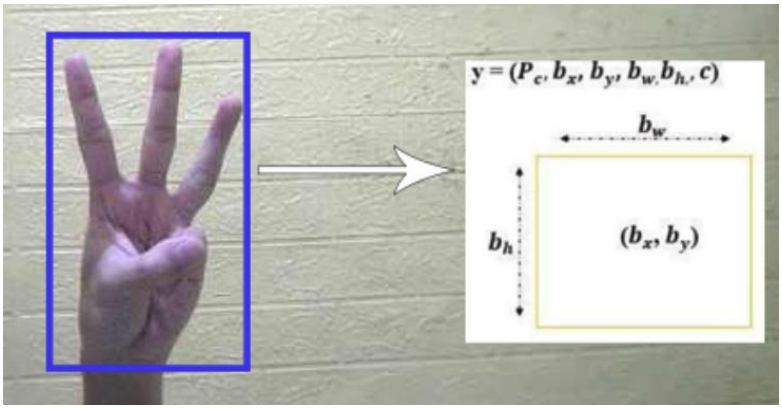
$IoU_{Pred}^{Truth}$= rasio tumpang tindih *Intersection Over Union* (IoU) antara kotak prediksi dan kotak *ground truth*.

$Pred$ = luas area dalam kotak prediksi.

$Truth$ = area dalam *ground truth*. Semakin besar nilai IoU, maka semakin tinggi tingkat akurasi pendeteksiannya.

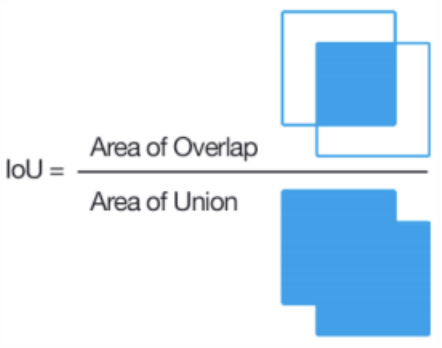
2. Nilai *confidence* dapat juga dihitung menggunakan nilai *Intersection Over Union* (IOU) antara *bounding box* yang diprediksi dengan *ground truth. Ground truth* merupakan *bounding box* yang dibuat manual saat proses pelabelan gambar. Jika tidak ada objek di dalam sel *grid*, maka nilai *confidence* adalah nol. Setiap sel *grid* juga memprediksi probabilitas C kelas kondisional.

3. Setiap *bounding box* memiliki 5 nilai informasi yaitu *x, y, w, h, c*. Dimana nilai *x* dan *y* merupakan koordinat titik tengah *bounding box* yang terprediksi, nilai *w* dan *h* merupakan ukuran lebar dan tinggi *bounding box*, dan nilai *c* atau *confidence* adalah representasi IOU antara *predicted box* dengan *ground truth box*. Pada Gambar 2.5 merupakan komponen *bounding box*.



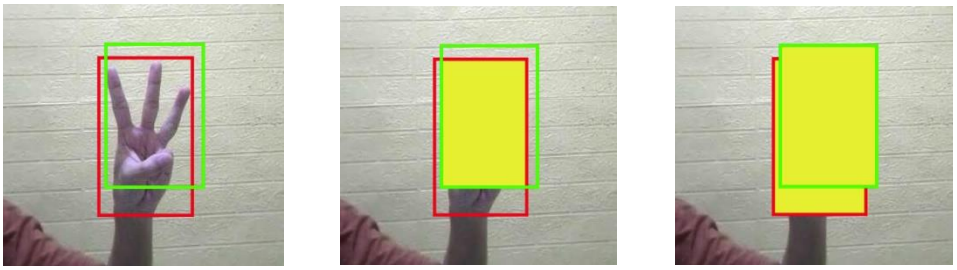

Gambar 2.5 Komponen *Bounding Box*

4. IoU dapat diperoleh dari perbandingan antara *ground truth box* dengan *predicted box* seperti pada Gambar 2.6.




Gambar 2.6 IoU (Hanafi, 2020)

5. Pada Gambar 2.7 objek yang dideteksi adalah tangan, dan terdapat dua kotak pada citra tersebut. Kotak berwarna merah adalah *ground truth box* yang didapatkan melalui proses pelabelan secara manual, yang tentunya akurasi lokalisasinya sangat hampir akurat, sedangkan kotak berwarna hijau adalah prediksi *bounding box* yang didapatkan melalui prediksi jaringan YOLO, dimana akurasi lokalisasinya tidak sebagus *ground truth box*. Untuk mendapatkan nilai IOU objek tangan, dilakukan perbandingan antara area tumpang tindih prediksi *bounding box* objek tangan dengan *ground truth box* objek mobil dan area gabungan prediksi *bounding box* dengan *ground truth box* objek tangan. Melalui perbandingan ini akan didapatkan nilai IOU yang berkisar antara 0 - 1, jika nilai IOU semakin mendekati angka satu artinya *bounding box* yang diprediksi mempunyai akurasi lokalisasi yang tinggi.



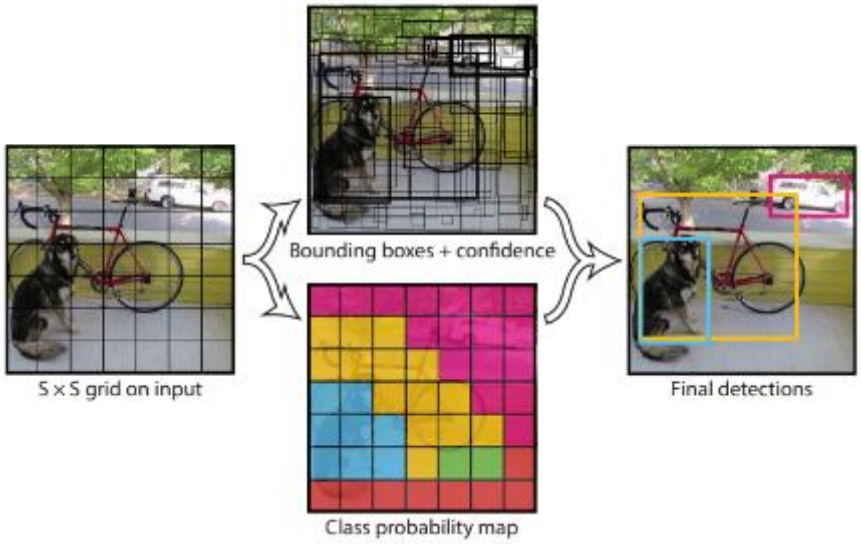
*Ground truth* dan *Predicted box* | *Intersection* | *Union*

Gambar 2.7 Contoh IoU

6. Untuk mendapatkan probabilitas sebuah objek muncul dalam sebuah *bounding box* dan seberapa cocok *bounding box* membatasi sebuah objek, YOLO mengalikan probabilitas kelas kondisional dan prediksi *confidence* sebuah *bounding box* seperti pada persamaan 2.2.

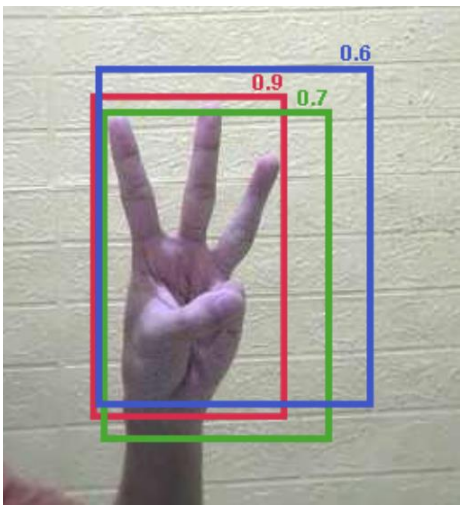
$$Pr(Class_i|Object) \times \Pr(Object) \times IoU_{Pred}^{Truth} = \Pr(class) \times IoU_{Pred}^{Truth} \quad \dots\dots\dots\dots (2.2)$$

Sehingga menghasilkan nilai *confidence* kelas secara spesifik pada tiap *bounding box*. Nilai tersebut menunjukkan *class* probabilitas yang muncul pada *bounding box* dan seberapa akurat *bounding box* memprediksi sesuai dengan objeknya. Pada Gambar 2.8 merupakan tahapan algoritma YOLO.

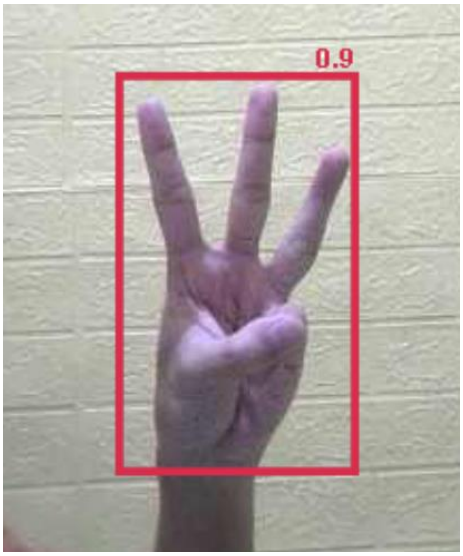



Gambar 2.8 Tahap Deteksi Objek pada YOLO (Redmon dkk, 2016)

Dalam proses deteksi objek, algoritma YOLO akan mendeteksi beberapa objek besar atau objek yang berdekatan dengan batas sel-sel *grid* dalam banyak *bounding box* yang akurasi lokalisasinya beragam dari nilai yang rendah (nilai IOU mendekati 0) sampai dengan nilai yang tinggi (nilai IOU mendekati 1). Hal ini menjadi masalah dan cara mengatasinya dapat menggunakan operasi *Non-maximal Suppression*. *Non-maximal Suppression* (NMS) adalah operasi pembuangan *bounding box* yang memiliki nilai IOU kurang dari *threshold* (nilai ambang batas) (Liang dkk, 2022). Sebagai contoh nilai *threshold* adalah 0,9, jika terdapat *bounding box* yang nilai IOU dibawah 0,9 maka akan dibuang. Pada Gambar 2.9 merupakan kondisi tanpa menggunakan NMS dan pada Gambar 2.10 merupakan kondisi setelah menggunakan NMS.




Gambar 2.9 Hasil Prediksi




Gambar 2.10 Hasil Pembuangan *Bounding Box* dibawah *threshold*

YOLOv4 merupakan versi terbaru yang dikembangkan oleh (Bochkovskiy, Wang dan Liao, 2020) yang menambahkan sejumlah implementasi tambahan untuk meningkatkan akurasi dan efisiensi dalam deteksi objek. Arsitektur ini menggunakan arsitektur backbone CNN 14 CSPDarknet53. Arsitektur ini berisi 162

lapisan dengan input gambar yang sudah dilakukan konfigurasi sesuai penelitian (Bochkovskiy dkk, 2020).

### 2.1.6. *Computer Vision*

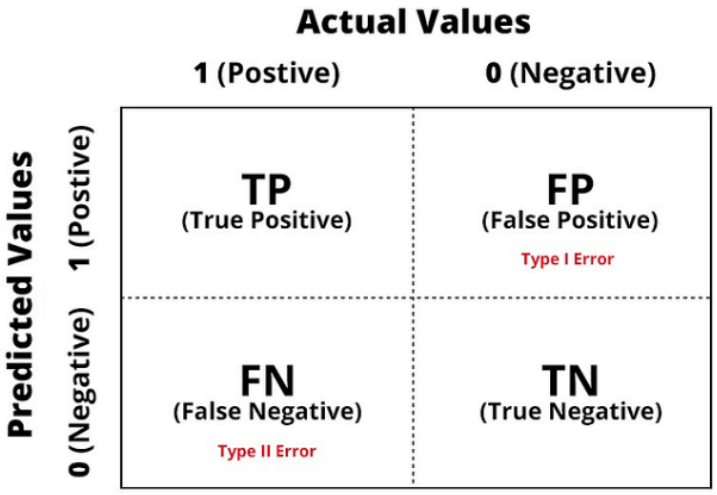
*Computer Vision* merupakan bagian dari teknik *Artificial Intelligence* yang memungkinkan komputer untuk melihat dan mengenali benda-benda di sekitarnya seperti manusia. *Computer Vision* terinspirasi dari kemampuan sistem penglihatan manusia (Andwiyan dkk, 2021). *Computer Vision* terdiri dari akuisisi citra (*image acquisition*), pengolahan citra (*image processing*), dan penerjemahan citra (*image understanding*) (Hasan, 2020).

### 2.1.7. Metrik Evaluasi

Dalam *machine learning* khususnya bidang *computer vision*, metrik dapat diartikan sebagai suatu nilai yang dapat digunakan untuk menggambarkan performa suatu model yang dihasilkan. Berikut ini merupakan metrik evaluasi pada kasus deteksi objek.

### 2.1.7.1.*Confusion Matrix*

*Confusion matrix* merupakan ringkasan hasil prediksi dari suatu masalah klasifikasi. Jumlah klasifikasi yang benar dan salah dikumpulkan dengan nilai yang dihitung kemudian dibagi ke setiap kelas. Jadi, *confusion matrix* tidak hanya memberikan informasi tentang kebenarannya oleh pengklasifikasi, tetapi juga jenis kesalahannya (Hanafi, 2020). Pada Gambar 2.11 merupakan tabel *confusion matrix*.




Gambar 2.11 *Confusion Matrix* (Nurhadiati dkk, 2022)

*Confusion matrix* dapat dilihat dengan beberapa ketentuan sebagai berikut .

a. *True Positive* (TP): aktual bernilai positif, dan diprediksi positif.
b. *True Negative* (TN): aktual bernilai negatif, dan diprediksi negatif.
c. *False Positive* (FP): aktual bernilai negatif, namun diprediksi positif.
d. *False Negative* (FN): aktual bernilai positif, namun diprediksi positif.

#### 2.1.7.2.*Precision*

*Precision* merupakan perbandingan antara *True Positive* (TP) dengan banyaknya data yang diprediksi *True Positive* (TP) dan *False Positive* (FP) seperti pada persamaan 2.3 (Nurhadiati dkk, 2022).

$$Precision = \frac{TP}{TP + FP} \quad \dots\dots\dots\dots\dots \quad (2.3)$$

#### 2.1.7.3.*Accuracy*

Akurasi menggambarkan keakuratan model dalam mengklasifikasikan dengan benar. Akurasi dapat dihitung menggunakan persamaan 2.4.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad \dots\dots\dots\dots\dots \quad (2.4)$$

#### 2.1.7.4.F1-*score*

F1-*score* menggambarkan perbandingan rata-rata *precision* dan *recall* yang dibobotkan. F1-score dapat dihitung menggunakan persamaan 2.5 (Nurhadiati dkk, 2022).

$$F1\ score = \frac{TP + TN}{TP + FP + FN + TN} \quad \dots\dots\dots\dots\dots \quad (2.5)$$

#### 2.1.7.5.*Recall*

*Recall* merepresentasikan tingkat keberhasilan model dalam menemukan kembali sebuah informasi. *Recall* merupakan perbandingan antara prediksi *True Positive* (TP) dengan banyaknya data yang diprediksi *True Positive* (TP) dan *False Negative* (FN) seperti pada persamaan 2.6 (Nurhadiati dkk, 2022).

$$Recall = \frac{TP}{TP + FN} \quad \dots\dots\dots\dots\dots \ (2.6)$$

### 2.1.7.6.*Average Precision* (AP)

*Average Precision* (AP) adalah cara untuk merangkum nilai *precision* dan *recall* menjadi satu nilai yang mewakili rata-rata semua *precision*. Menggunakan *loop* yang melewati semua *precision* dan *recall*, perbedaan *recall* saat ini dan berikutnya akan dihitung dan kemudian dikalikan dengan *precision* saat ini (Nurhadiati dkk, 2022). AP dihitung menggunakan persamaan 2.7 kemudian persamaan 2.8.

$$AP = \sum(recall_{n+1} - \ recall_n)\ precision_{interp}(recall_{n+1}) \ \dots\dots\dots \ (2.7)$$

$$P_{interp}(r_{n+1}) = \max p(\tilde{r});\ \ \tilde{r} \geq r_{n+1} \ \dots\dots\dots \ (2.8)$$

### 2.1.7.7.*Mean Average Precision* (mAP)

*Mean Average Prescision* (mAP) merupakan nilai rata-rata *Average Precision* (AP) dari suatu objek pada setiap kelas (Nurhadiati dkk, 2022). Nilai AP diperoleh dari perhitungan *precision* pada persamaan 2.3 dan recall pada persamaan 2.6, lalu mAP dihitung menggunakan persamaan 2.9. Dimana $N$ adalah jumlah kelas dan $AP_i$ adalah nilai AP setiap kelas.

$$\text{mAP} = \sum_{i=1}^{N} \frac{AP_i}{N} \times 100\% \quad \dots\dots\dots\dots\dots \ (2.9)$$

### 2.1.8. *Prototype*

*Prototype* merupakan suatu model kerja skala kecil dari keseluruhan sistem yang berisi komponen-komponen dari sistem baru yang paling menarik bagi pengguna. Dengan kata lain *prototype* adalah suatu versi sistem yang disediakan bagi pengembang dan calon pengguna yang memberikan suatu gambaran tentang sistem yang akan dibangun dan dapat berfungsi jika telah disusun dalam bentuk yang sempurna (Kurniati, 2021).

### 2.1.9. *Lift*

*Lift* atau *elevator* merupakan suatu teknologi dalam bangunan bertingkat yang dapat memudahkan manusia untuk berpindah dari satu lantai ke lantai lainnya. Prinsip kerja dasar dari sebuah *lift* adalah penggunaan katrol dan energi listrik sebagai sumber daya penggerak untuk menggerakan atau berpindah lantai. Saat ini penggunaan lift masih menggunakan tombol, dimana setiap pengguna *lift* dapat menekan tombol sesuai lantai tujuannya (Nugraha dkk, 2015).

## 2.2. Penelitian Terkait

### 2.2.1. *State of The Art*

Banyak penelitian yang mengkaji terkait deteksi *hand gesture* sebagai kontrol atau pengendali perangkat. Namun, masing-masing penelitian memiliki metode dan hasil yang berbeda. Berikut ini merupakan penelitian terkait dengan penelitian yang dilakukan disajikan pada Tabel 2.1.

Tabel 2.1 Penelitian Terkait

| No. | Peneliti | Objek Penelitian | Metode | Hasil / Parameter yang diuji |
|---|---|---|---|---|
| 1. | (Huu dan Phung Ngoc, 2021) | *Hand Gesture* sebagai kontrol sistem robot | SVM dan HOG | Akurasi 90% dari 1000 *training data* |
| 2. | (Yunita dan Setyati, 2019) | *Hand Gesture* sebagai pengganti mouse komputer | Convex Hull | Akurasi 68% dari 75 kali percobaan |
| 3. | (Hidayatullah, Setyawan dan Akbar, 2018) | *Hand Gesture* sebagai kontrol *Drone Quadcopter* | Complementary Filter | Delay sistem 0,67 detik |
| 4. | (Taban dkk, 2020) | *Hand Gesture* sebagai kontrol *switch* LED | Viola Jones | Delay sistem 0,52 detik |
| 5. | (Tarvekar, 2019) | *Hand Gesture* pada mobil tanpa sentuh | Multiclass SVM | Akurasi 92% dari 7 kali percobaan |
| 6. | (Alvin dkk, 2021) | *Hand Gesture* untuk *American Sign Language* | K-Nearest Neighbor | Akurasi 94% |
| 7. | (Tanmaie dan Rao, 2020) | *Hand Gesture* berbasis *Deep Learning* | YOLOv2 | Akurasi rata-rata 100% dari 200 gambar tangan dan 97,46% dari 750 gambar tangan dengan *noise* |

| No. | Peneliti | Objek Penelitian | Metode | Hasil / Parameter yang diuji |
|---|---|---|---|---|
| 8. | (Bose dan Kumar, 2019) | *Hand Gesture* berbasis *Deep Learning* | Faster R-CNN | Akurasi rata-rata *precision* 0,794, *recall* 0,833, dan F1 *score* 0,813 |
| 9. | (Asri dkk, 2019) | *Hand Gesture* untuk bahasa isyarat Malaysia | YOLOv3 | Akurasi 63% dari hasil *training* dengan 7000 *step*, dan 72% dari pengujian aktual |
| 10. | (Su dkk, 2021) | *Hand Gesture* sebagai kontrol peralatan rumah tangga | K-Nearest Neighbor | Akurasi 91% dengan waktu pemrosesan 30 detik |
| 11. | (Chen dan Huang, 2021) | *Hand Gesture* sebagai kontrol lengan robot | YOLOv4 | Mean Average Precission (mAP) sebesar 99,93% |
| 12. | (Baek dan Lee, 2022) | *Hand Gesture* sebagai kontrol *traffic* | Recurrent Neural Network (RNN) | Akurasi 91% dengan latar belakang yang berbeda |
| 13. | (Huu, Minh dan The, 2020) | *Hand Gesture* untuk aplikasi *Smart Home* | Artificial Neural Network (ANN) | Akurasi rata-rata 92,6% untuk gambar dan 91,5% untuk video |

### 2.2.2. Matriks Penelitian

Berikut ini merupakan matriks penelitian terkait deteksi *hand gesture* sebagai kontrol perangkat yang disajikan pada Tabel 2.2.

Tabel 2.2 Matriks Penelitian

| No. | Peneliti | Objek Penelitian | Metode | | | | | | | | | | | | Parameter | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *SVM* | *HOG* | *Convex Hull* | *Complementary Filter* | *Viola Jones* | *Multiclass SVM* | *K-Nearest Neighbor* | *YOLOv2* | *Faster R-CNN* | *YOLOv3* | *RNN* | *ANN* | *YOLOv4* | *Akurasi* | *Waktu* | *mAP* |
| 1. | (Huu dan Phung Ngoc, 2021) | *Hand Gesture* sebagai kontrol sistem robot | ✓ | ✓ | | | | | | | | | | | | ✓ | | |
| 2. | (Yunita dan Setyati, 2019) | *Hand Gesture* sebagai pengganti mouse komputer | | | ✓ | | | | | | | | | | | ✓ | | |

| No. | Peneliti | Objek Penelitian | Metode | | | | | | | | | | | | Parameter | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *SVM* | *HOG* | *Convex Hull* | *Complementary Filter* | *Viola Jones* | *Multiclass SVM* | *K-Nearest Neighbor* | *YOLOv2* | *Faster R-CNN* | *YOLOv3* | *RNN* | *ANN* | *YOLOv4* | *Akurasi* | *Waktu* | *mAP* |
| 3. | (Hidayatullah, Setyawan dan Akbar, 2018) | *Hand Gesture* sebagai kontrol *Drone Quadcopter* | | | | ✓ | | | | | | | | | | | ✓ | |
| 4. | (Taban dkk, 2020) | *Hand Gesture* sebagai kontrol *switch* LED | | | | | ✓ | | | | | | | | | | ✓ | |
| 5. | (Tarvekar, 2019) | *Hand Gesture* pada mobil tanpa sentuh | | | | | | ✓ | | | | | | | | ✓ | | |

| No. | Peneliti | Objek Penelitian | Metode | | | | | | | | | | | | Parameter | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *SVM* | *HOG* | *Convex Hull* | *Complementary Filter* | *Viola Jones* | *Multiclass SVM* | *K-Nearest Neighbor* | *YOLOv2* | *Faster R-CNN* | *YOLOv3* | *RNN* | *ANN* | *YOLOv4* | *Akurasi* | *Waktu* | *mAP* |
| 6. | (Alvin dkk, 2021) | *Hand Gesture* untuk *American Sign Language* | | | | | | | ✓ | | | | | | | ✓ | | |
| 7. | (Tanmaie dan Rao, 2020) | *Hand Gesture* berbasis *Deep Learning* | | | | | | | | ✓ | | | | | | ✓ | | |
| 8. | (Rubin Bose dan Sathiesh Kumar, 2019) | *Hand Gesture* berbasis *Deep Learning* | | | | | | | | | ✓ | | | | | ✓ | | |

| No. | Peneliti | Objek Penelitian | Metode | | | | | | | | | | | | Parameter | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *SVM* | *HOG* | *Convex Hull* | *Complementary Filter* | *Viola Jones* | *Multiclass SVM* | *K-Nearest Neighbor* | *YOLOv2* | *Faster R-CNN* | *YOLOv3* | *RNN* | *ANN* | *YOLOv4* | *Akurasi* | *Waktu* | *mAP* |
| 9. | (Asri dkk, 2019) | *Hand Gesture* untuk bahasa isyarat Malaysia | | | | | | | | | | ✓ | | | | ✓ | | |
| 10. | (Su dkk, 2021) | *Hand Gesture* sebagai kontrol peralatan rumah tangga | | | | | | | ✓ | | | | | | | ✓ | ✓ | |
| 11. | (Chen dan Huang, 2021) | *Hand Gesture* sebagai kontrol lengan robot | | | | | | | | | | | | | ✓ | | | ✓ |

| No. | Peneliti | Objek Penelitian | Metode | | | | | | | | | | | | Parameter | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *SVM* | *HOG* | *Convex Hull* | *Complementary Filter* | *Viola Jones* | *Multiclass SVM* | *K-Nearest Neighbor* | *YOLOv2* | *Faster R-CNN* | *YOLOv3* | *RNN* | *ANN* | *YOLOv4* | *Akurasi* | *Waktu* | *mAP* |
| 12. | (Baek dan Lee, 2022) | *Hand Gesture* sebagai kontrol *traffic* | | | | | | | | | | | ✓ | | | ✓ | | |
| 13. | (Huu, Minh dan The, 2020) | *Hand Gesture* untuk aplikasi *Smart Home* | | | | | | | | | | | | ✓ | | ✓ | | |