Jurnal Rekayasa Elektrika Vol. 18, No. 2, Juni 2022, hal. 115-124
ISSN. 1412-4785; e-ISSN. 2252-620X, Terakreditasi Dikbudristekdikti No. 105/KPT/2022
DOI: 10.17529/jre.v18i2.25147

Deteksi Gestur Tangan berbasis Pengolahan Citra

Abdullah Sani dan Suci Rahmadinni
Politeknik Negeri Batam
Jalan Ahmad Yani Batam Centre, 29461
e-mail: sani@polibatam.ac.id

***Abstrak*—Bahasa isyarat tangan adalah media komunikasi untuk penyandang disabilitas (tuna rungu dan tuna wicara). Namun, dalam kehidupan sehari–hari tidak jarang penyandang disabilitas harus berkomunikasi dengan orang normal yang tidak mengerti dengan bahasa isyarat. Masalah tersebut dapat diatasi dengan bantuan penerjemah atau orang normal belajar bahasa isyarat melalui media yang ada seperti video. Namun, cara ini akan menghabiskan banyak biaya dan waktu. Pada penelitian ini, penulis merancang sebuah sistem untuk mendeteksi gerakan tangan berbasis *image processing*. Metode yang digunakan adalah menggunakan algoritma *You Only Look Once* (YOLO-v3). Algoritma YOLO-*version* 3 dapat mendeteksi dan mengklasifikasi objek sekaligus tanpa di pengaruhi oleh intensitas cahaya dan *background* dari objek. Algoritma ini merupakan metode *deep learning* yang lebih akurat dari pada metode *deep learning* lainnya. Diharapkan dengan adanya penelitian ini, mampu membuat sebuah sistem deteksi pengenalan gesture tangan sebagai bahasa isyarat sehingga dapat membantu komunikasi antara orang normal dengan disabilitas (tuna rungu dan tuna wicara). Dari hasil pengujian yang telah dilakukan, maka didapatkan sistem dapat mendeteksi dan mengklasifikasi gesture tangan dengan *background*, intensitas cahaya, dan jarak yang berbeda–beda dengan tingkat akurasi diatas 90%.**

Kata kunci: ***deep learning, bahasa isyarat, tracking, YOLO-version* 3**

***Abstract*—Hand sign language is a medium of communication for people with disabilities (deaf and speech impaired). However, in social practice, persons with disabilities may have to communicate with non-disable persons who do not understand sign language. These problems can be overcome with the help of translators or normal people learning sign language through existing media such as videos. Unfortunately, this method will probably cost a lot of money and time. In respons to this issue, the present study designed a sistem to detect hand gestures based on image processing. The method used is the You Only Look Once (YOLO) algorithm. The YOLO algorithm can detect and classify objects at once without being influenced by the light intensity and background of the object. This algorithm is a deep learning method that is more accurate than other deep learning methods. From this research, the system can detect and classify hand gestures with different backgrounds, light intensity, and distances with an accuracy rate above 90%.**

Keywords: ***deep learning, sign language, tracking, YOLO-v3***

I. Pendahuluan

Bahasa isyarat adalah media komunikasi yang digunakan oleh penyandang disabilitas dalam berkomunikasi [1]. Umumnya pada bahasa isyarat, *gesture* tangan digunakan untuk menyampaikan informasi alih-alih menggantikan suara. Agar pesan dalam berkomunikasi dapat disampaikan dengan baik, maka subjek yang sedang berkomunikasi harus mengerti dengan bahasa yang digunakan. Dalam kehidupan sehari-hari, kebanyakan orang berkomunikasi dengan menggunakan bahasa verbal yaitu lisan dan tulisan. Untuk disabilitas khususnya tuna rungu dan tuna wicara yang tidak dapat berbicara maupun mendengar, berkomunikasi dengan menggunakan bahasa isyarat. Perbedaan cara berkomunikasi menyebabkan komunikasi antara orang normal dan tuna rungu serta tuna wicara tidak berjalan dengan baik. Hal ini dikarenakan orang normal yang belum mengerti arti dari pola gerakan tangan dalam bahasa isyarat.

Penelitian yang telah dilakukan untuk memecahkan masalah tersebut adalah dengan menggunakan kamera sebagai pendeteksi gerakan tangan manusia menggunakan metode *Backpropogation* [2]. Penelitian lain yaitu deteksi bahasa isyarat menggunakan segmentasi *YCbCr* [3] dan jaringan syaraf tiruan [4]. Penelitian lainnya yang telah dilakukan adalah *tracking* dan pengenalan pola gerak tangan berbasis penginderaan visual yang menggunakan pengolahan citra untuk mendeteksi pola gerak tangan menggunakan kamera [5]. Namun, penelitian – penelitian tersebut memiliki kelemahan yaitu kurangnya tingkat keakurasian dalam pendeteksian karena dipengaruhi oleh kondisi pencahayaan yang berbeda-beda dan *background* dari pola gerak tangan harus berwarna putih.

Untuk mengatasi pendeteksian agar tidak dipengaruhi oleh kondisi pencahayaan dan *background* dari *gesture* tangan, maka pada penelitian ini akan dikembangkan sistem deteksi pola gerak tangan tanpa harus dipengaruhi oleh *background* pola tangan dan kondisi pencahayaan

Received 08 March 2022; Revised 21 March 2022; Accepted 22 March 2022

yang berubah – ubah. Metode yang akan digunakan adalah *You Only Look Once* (YOLO-*version 3*). *YOLO* adalah metode untuk mendeteksi objek yang perkenalkan oleh Redmon [6]. Algoritma ini mendeteksi dan mengklasifikasi objek dengan performa yang lebih baik daripada metode sebelumnya. *YOLO* menggunakan metode *anchor-based detection* dimana *anchor box* akan di-*generate* dalam sekali proses untuk setiap *frame* sehingga tingkat akurasinya relatif lebih baik. Dan dapat terbentuk sebuah sistem pendeteksian dan pengklasifikaisan *gesture* tangan akurat yang dapat membatu komunikasi antara orang normal dengan tuna rungu atau tuna wicara.

II. Studi Pustaka

A. Hand Gesture Detection dan Bahasa Isyarat

Bahasa isyarat adalah media komunikasi utama yang digunakan di seluruh dunia oleh penderita disabilitas yaitu tuna rungu dan tuna wicara. Dalam berkomunikasi, bahasa isyarat menggunakan gerak tubuh, gerakan bibir, *gesture* tangan dan ekspresi wajah untuk mengganti suara. *Gesture* tangan ada dua kategori, yaitu statik dan dinamis. *Gesture* tangan statik adalah pengenalan *input* bahasa isyarat yang bersifat diam atau statik. Karena hanya diam, maka isyarat ini digunakan untuk mengenal pose tangan maupun abjad. Sedangkan isyarat dinamis adalah pengenalan masukan bahasa isyarat yang bersifat bergerak [7].

American Sign Language (ASL) adalah nama bahasa isyarat di Amerika, sementara Sistem Bahasa Isyarat Indonesia (SIBI) dan Bahasa Isyarat Indonesia (BISINDO) adalah dua bahasa isyarat yang digunakan di Indonesia. Dua bahasa isyarat yang digunakan di Indonesia terdapat perbedaan dalam pengaplikasiannya. SIBI digunakan untuk mengganti kosa kata dalam Bahasa Indonesia ke dalam bahasa isyarat Sedangkan BISINDO adalah isyarat gerak tangan sesuai bahasa tubuh nurani penyandang disabilitas yang diterjemahkan ke dalam bahasa isyarat.

Penelitian telah dilakukan oleh Kanchan Dabre dan Surekha Dholay [8], metode yang digunakan adalah *image processing* untuk identifikasi bentuk tangan, *Haar cascade classifier* untuk menerjemahkan bahasa isyarat, dan *Speech Synthesizer* untuk mengkonversi teks ke suara. Penelitian ini menghasilkan sistem yang mendeteksi karakteristik dari bentuk tangan dalam bentuk teks kemudian di *convert* ke dalam suara dengan tingkat akurasi kecepatan 92,68% [8]. Penelitian selanjutnya dilakukan oleh Lionel Pigou [9] dengan menggunakan metode *Convolutional Neural networks (CNNs)*, menghasilkan sistem yang dapat mengenali 20 isyarat Italia.

Penelitian lainnya dilakukan oleh Siska Eka Octaviani tentang *Tracking* dan Pengenalan Pola Gerak Tangan Berbasis Penginderaan Visual dengan menggunakan metode pengolahan citra dan jaringan saraf tiruan (*Neural network*). Menghasilkan Sistem *tracking* dan pengenalan pola gerak tangan yang mampu memisahkan objek tangan dengan *background* dan mengenali *gesture* secara *real time*.

Penelitian terdahulu di atas masih memiliki kekurangan yaitu deteksi *gesture* tangan yang dipengaruhi oleh intensitas cahaya dan *background* dari tangan sehingga mempengaruhi keakuratan dari pendeteksian. Pada penelitian ini, penulis merancang sebuah sistem untuk mendeteksi *gesture* tangan tanpa dipengaruhi oleh intensitas cahaya dan *background* dari tangan dengan menggunakan *image processing*. Metode *image processing* yang digunakan adalah algoritma *Y*ou Only Look Once (YOLO-v3) yang dapat mendeteksi dan mengklasifikasi objek sekaligus tanpa dipengaruhi oleh intensitas cahaya dan *background* dari objek sehingga sistem dapat mendeteksi objek dengan akurasi yang lebih baik. You Only Look Once menerapkan *single neural network* yang membagi gambar menjadi beberapa sel grid dan menghasilkan probabilitas sel sehingga sistem dapat memprediksi *bounding box* yang mencangkup beberapa wilayah grid dan memilih probabilitas yang terbaik [10].

B. Deep learning

Deep learning merupakan bagian dari *machine learning* yang algoritmanya mirip dengan sistem saraf manusia dimana setiap neuron terhubung satu sama lain dan saling menyampaikan informasi. Struktur dan fungsi *deep learning* terinspirasi dari *artificial neural network* dimana data dapat diolah dalam jumlah besar pada sebuah komputer dengan memperdalam atau memperbanyak lapisan jaringan (*layer*).

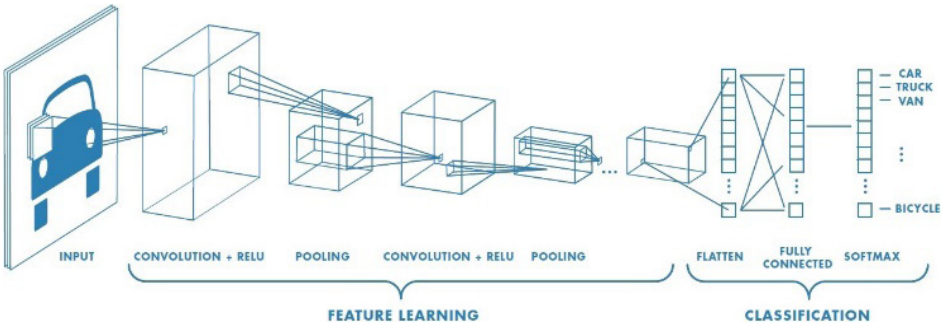
Semakin dalam atau banyak jumlah lapisan jaringan, maka komputer akan semakin cepat dalam mengolah informasi dari data yang diterima. Data yang diterima direpresentasikan pada semua lapisan jaringan untuk diolah, dari yang paling sederhana sampai yang paling kompleks. Hasil dari representasi digunakan untuk memprediksi objek yang dideteksi dengan mencoc*ok*kan data yang sudah dipelajari. Dengan menggunakan *deep learning* maka akan menciptakan komputer yang memiliki kemampuan olah data yang akurat, sehingga semakin banyak data yang diberikan maka akan semakin baik hasil dari pembelajarannya [11], [12].

1. *Convolutional Neural network (CNN)*

Convolutional Neural network adalah jenis *neural network* yang biasa digunakan pada data berupa gambar. *CNN* tidak jauh berbeda dengan *neural network* biasanya, hanya saja *CNN* terdiri dari neuron yang memiliki *weight, bias,* dan *activation* seperti ditunjukkan pada Gambar 1 [13], [14], [15].

2. *You Only Look Once* (*YOLO-v3*)

Salah satu metode *deep learning* untuk mendeteksi objek yaitu *You Only Look Once* (*YOLO*) yang diperkenalkan oleh Redmon, dkk. *YOLO* memiliki kecepatan pendeteksian objek yang lebih cepat dibandingkan metode *deep learning* lainnya. Metode ini memiliki performa hingga 45 FPS dalam mendeteksi objek secara *real-time* [16], [17]. *YOLO* menggunakan



Gambar 1. Convolutional neural network [13]

convolutional neural network untuk mendeteksi objek. *YOLO* menerapkan *single neural network* ke seluruh citra. *YOLO-v3* adalah algoritma yang digunakan untuk mendeteksi objek yang menggunakan *darknet*-53 sebagai *feature framework*. Pada awalnya *YOLO-v3* mempunyai 53-*layer* CNN kemudian saat pendeteksian 53-*layer* lagi ditumpuk di atasnya sehingga menjadi 106 *layer*. *YOLO-v3* menggunakan *multilabel classification* untuk memprediksi kelas pada tiap *bounding box*.

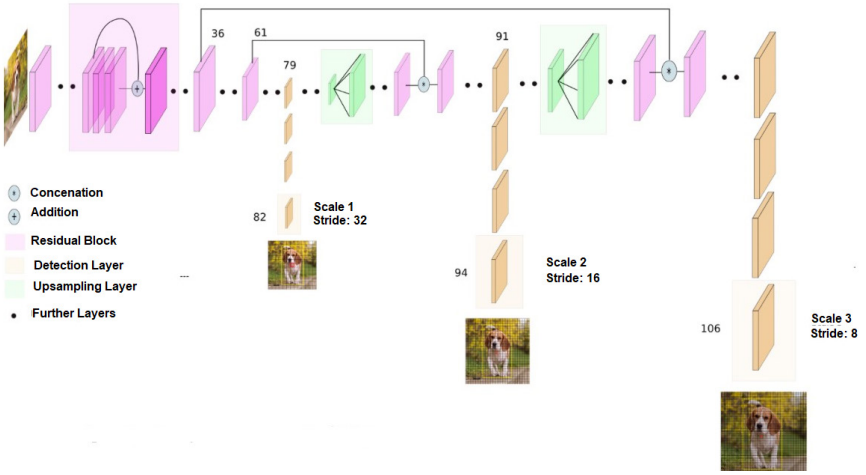
Hal ini karena setiap *bounding box* mungkin mempunyai lebih dari satu kelas ataupun tidak mempunyai kelas sama sekali. Untuk *input image* yang 416x416 3 skala yang digunakan pada *multilabel classification* yaitu 52x52 untuk objek yang kecil, 26x26 untuk objek menengah dan 13x13 untuk objek yang besar.

YOLO-v3 network bertujuan untuk memprediksi *bounding box* dengan probabilitas kelas tempat objek tersebut berada. Arsitektur *YOLO-v3* diperlihatkan pada Gambar 2. Dalam mendeteksi objek *YOLO* menerapkan jaringan saraf tunggal. Pada awalnya *YOLO* akan membagi citra menjadi beberapa *grid* S×S. Setiap sel *grid* memprediksi kotak pembatas *B* dan probabilitas kelas *C* dari objek. *Bounding box* menunjukkan posisi kelas dari objek dan dikaitkan juga dengan jumlah *anchors* yang digunakan. Setiap *bounding box* mempunyai *attributes* 5+*C*, dimana 5 mengartikan 5 koordinat *bounding box* yaitu titik tengah (b_x,b_y), tinggi (b_h), lebar (b_w), dan *confidence score* dan *C* adalah jumlah kelas. *Output* dari meneruskan gambar ke jaringan *single forward pass convolution* adalah tensor 3D. *Output*nya akan terlihat seperti Gambar 3.

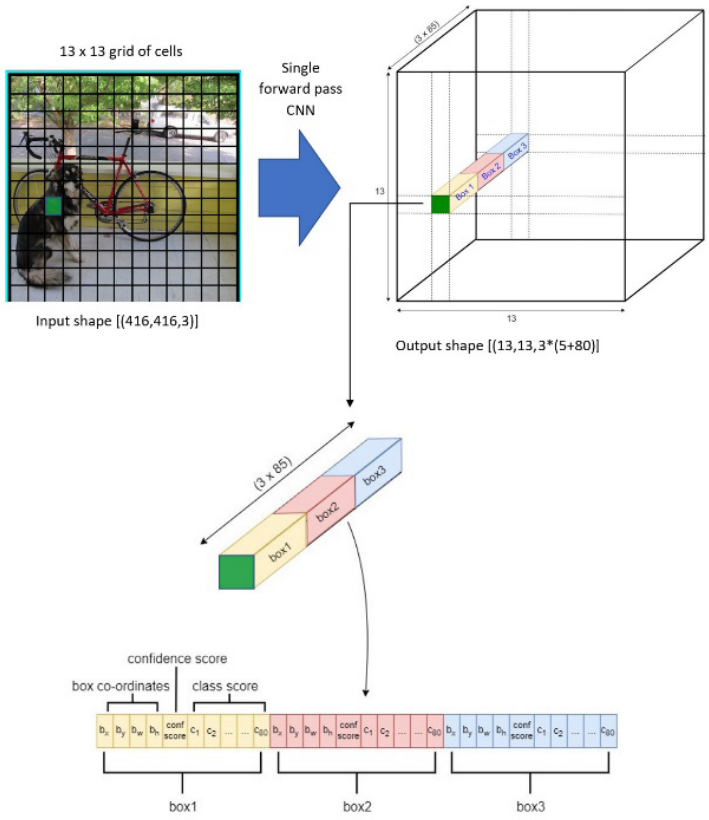
Pada Gambar 3 *input* gambar dibagi menjadi 13x13 *grid* yang mana setiap *grid* akan bertanggung jawab untuk memprediksi objek yang ada didalamnya. Karena *YOLO*-v3 memakai *multiscale detection feature* seperti diperlihatkan pada Gambar 4, deteksi *kernel* dengan tiga ukuran berbeda diterapkan di tiga tempat berbeda, sehingga nilai *B*=3. Pada penelitian ini *YOLO*-v3 di *training* dengan 10 kelas objek, maka *C*=10. Jadi *output tensor* 3D menjadi dimensi (13, 13.3 ×(10+5)).

Feature matrix persegi didefinisikan ke sebuah *grid*, namun tidak semua objek yang dideteksi berbentuk persegi (biasanya persegi panjang). Untuk itu menggunakan *anchor boxes*. *Anchor boxes* adalah kotak yang telah ditentukan sebelumnya yang mempunyai rasio aspek. Rasio aspek ditentukan sebelumnya bahkan sebelum *training* dengan menjalankan *K-means clustering* pada seluruh dataset. *Anchor boxes* akan diletakkan ke sel *grid* dengan titik tengah sama dengan titik tengah *bounding box*. *YOLO*-v3 menggunakan 3 *anchor boxes* untuk setiap skala deteksi sehingga jumlah semua *anchor boxes* menjadi 9.

YOLO-v3 memprediksi 4 koordinat untuk setiap *bounding box*, yaitu t_x, t_y, t_w, t_h seperti terlihat pada Gambar 5. Jika suatu sel batasnya di sudut kiri atas (C_x, C_y) dan lebar dan tinggi *anchor boxes* adalah P_w dan P_h, maka rumus untuk memprediksi *bounding box* adalah sebagai



Gambar 2. Arsitektur YOLO-v3 [18]



Gambar 3. Cara kerja YOLO dengan grid 13x13 [18]

berikut:

$$b_x = \sigma(t_x) + (c_x) \quad (1)$$

$$b_y = \sigma(t_y) + (c_y) \quad (2)$$

$$b_w = p_e e^{t_w} \quad (3)$$

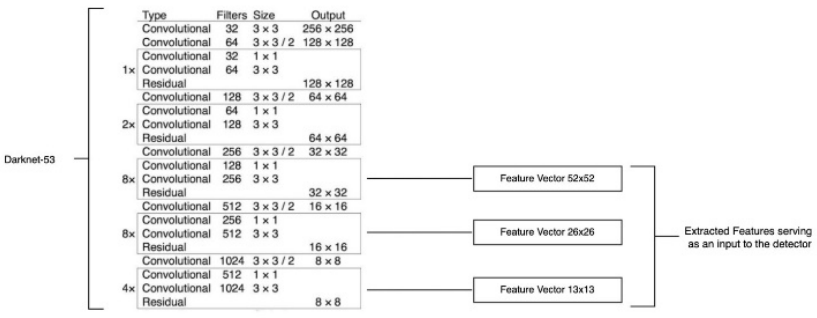
$$b_h = p_h e^{t_h} \quad (4)$$

Keterangan Gambar 5:
b_x = Koordinat tengah x *bounding box* yang diprediksi
b_y = Koordinat tengah y *bounding box* yang diprediksi
b_w = Koordinat lebar *bounding box* yang diprediksi
b_h = Koordinat tinggi *bounding box* yang diprediksi

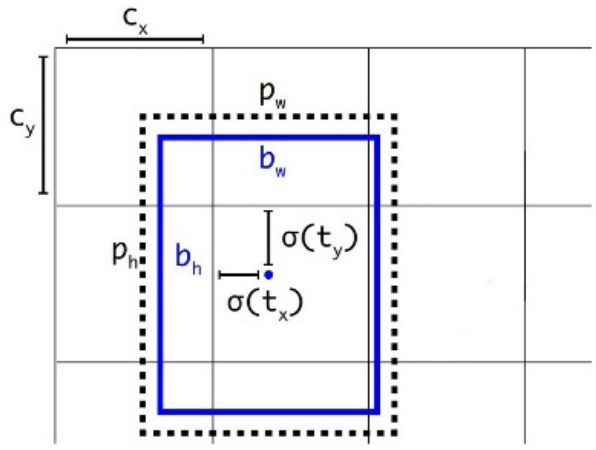
III. Metode/desain

A. Data Image

Data *image* berisi data gambar yang akan diolah pada saat *training*. Pengambilan data dilakukan dengan mengumpulkan data citra yang diambil dengan menggunakan webcam. Jarak objek dengan kamera saat pengambilan data adalah 30 cm sampai 80 cm. Kondisi ruangan saat pengambilan data menggunakan bantuan cahaya lampu yang intensitasnya disesuaikan menggunakan bantuan *dimmer switch*. Data yang diambil berupa isyarat *gesture* tangan yang akan dideteksi dan menggunakan *background* biru, putih, dan kuning. Jumlah data yang diambil sebanyak 1500 data yang terdiri dari 150



Gambar 4. Multi-Scale Feature Extraction untuk gambar 416x416 [18]



Gambar 5. Bounding box dengan Anchor box dan prediksi lokasi

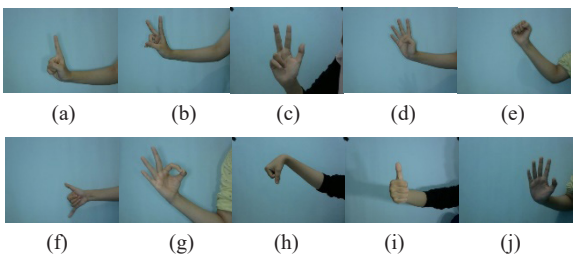
data disetiap *gesture* dengan menggunakan 3 *background* yang berbeda (biru, putih, dan kuning). Kondisi Gerakan tangan dapat dilihat pada Gambar 6.

B. -Alur Kerja Sistem

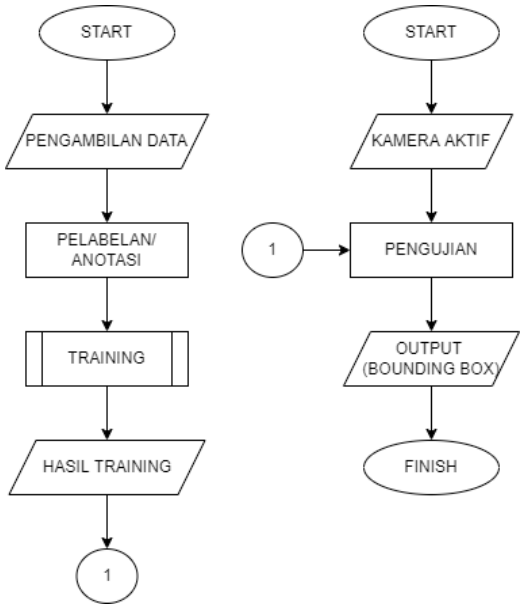
Alur kerja sistem diperlihatkan pada Gambar 7. Pada tahap awal yaitu akuisisi data berupa pengambilan dataset yang digunakan untuk proses *training*. Pengambilan data dilakukan dengan mengumpulkan data citra yang diambil dengan menggunakan webcam. Selanjutnya dataset yang telah diambil akan dilabeli atau anotasi yaitu memberi nama pada objek sesuai dengan kelasnya masing – masing. Proses pelabelan menggunakan *YOLO Mark* yang di unduh *melalui repository github.* Setelah proses pengunduhan selesai, terlebih dahulu kita mengatur konfigurasi untuk objek data dan objek *names* pada *YOLO* Mark. Objek *names* berisi nama kelas dan ID dari data yang akan dilabeli.

Hasil dari pelabelan data akan disimpan ke dalam file dengan format *.txt* yang didalamnya terdapat nama kelas, titik tengah sumbu x, titik tengah sumbu y, panjang *bounding box* dan lebar *bounding box*.

Setelah semua data selesai dilabel, maka dilanjutkan ke proses *training* data yang bertujuan untuk melatih sistem dengan cara mengolah data citra dan anotasi yang telah dilakukan sebelumnya sehingga terbentuk karakteristik dari masing–masing kelas agar menjadi bahan pertimbangan sistem dalam membuat sebuah keputusan atau memprediksi. Proses *training* dilakukan



Gambar 6. (a) Gesture satu, (b) Gesture dua, (c) Gesture tiga, (d) Gesture empat, (e) Gesture diam, (f) Gesture call, (g) Gesture ok, (h) Gesture no, (i) Gesture yes, (j) Gesture hello



Gambar 7. Alur kerja sistem

dengan menggunakan *framework neural network darknet*. Langkah pertama adalah mengunduh *file darknet* yang didalamnya telah ada file konfigurasi dari github dan mengatur konfigurasi untuk *training*nya sesuai dengan Tabel 1.

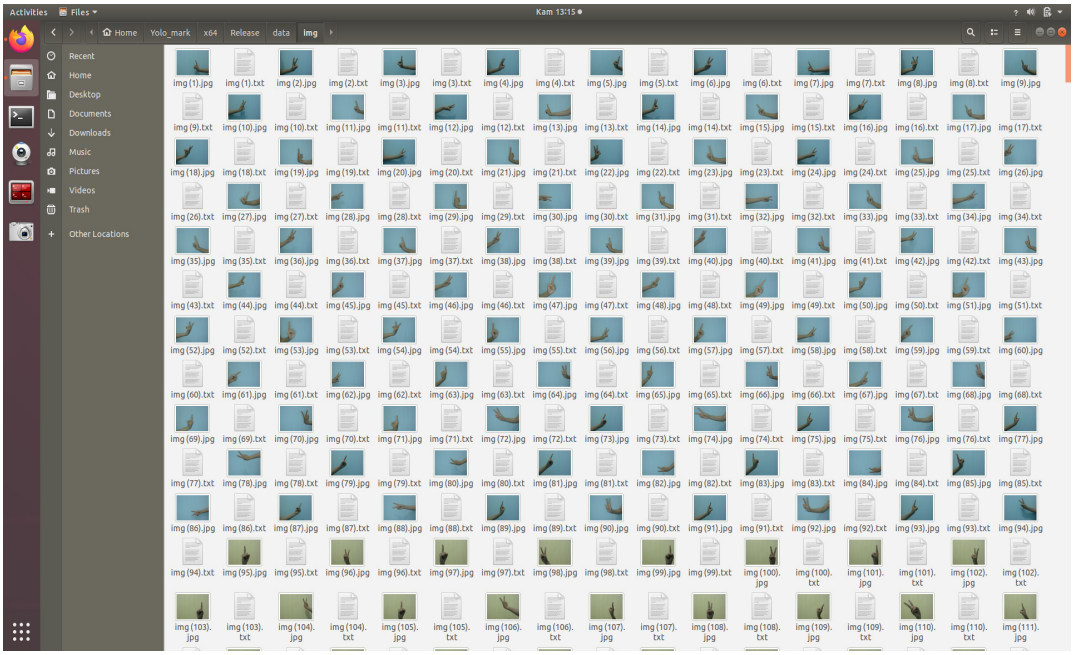
Setelah mengatur konfigurasi pada *darknet* selanjutnya kita melakukan konfigurasi untuk *weight* yang akan digunakan. Karena menggunakan *YOLO-v3* maka ubah konfigurasi untuk *YOLO-v3.* Konfigurasi yang diatur terdapat pada Tabel 2.

Batch menunjukkan jumlah gambar yang akan diproses sebelum *network weight* diperbaharui. *Subdivisions* digunakan untuk menangani sebagian kecil *batch size* sekaligus di *GPU. Max_batches* adalah batas iterasi untuk melakukan *training*. *Height* dan *width* menunjukkan dimensi dari gambar input yang akan di *training*. *Classes* merupakan jumlah kelas yang akan dideteksi.

Setelah mengatur konfigurasi untuk *darknet* dan *weight YOLO*-v3, buat folder baru di *darknet* kemudian pindahkan folder gambar yang telah di label/anotasi sebelumnya seperti terlihat pada Gambar 8. Lalu pindahkan juga *obj. names* dan *train.txt* ke foder *darknet*/data. Setelah itu pindahkan *obj.data* ke folder *darknet*/*cfg*. Dan terakhir jalankan *training* dengan menggunakan *command* seperti Gambar 9.

Tabel 1. Konfigurasi pada Darknet

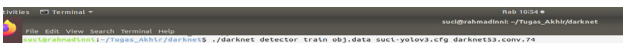
Jenis Konfigurasi	Keterangan
Load Model	darknet53
Load Weight	YOLO-v3
GPU	Yes
CUDNN	Yes
OPENCV	Yes
DEBUG	Yes
LIBSO	Yes



Gambar 8. Hasil gambar yang sudah dilabel/anotasi

Tabel 2. Konfigurasi YOLO-v3

Konfigurasi	Nilai
Batch	4
Subdivisions	1
Width	416
Height	416
Max_batches	500200
Classes	10
Filter	45
Anchors	(10x13), (16x30), (33x23), (30x61), (62x45), (59x119), (116x90), (156x198), (373x326)



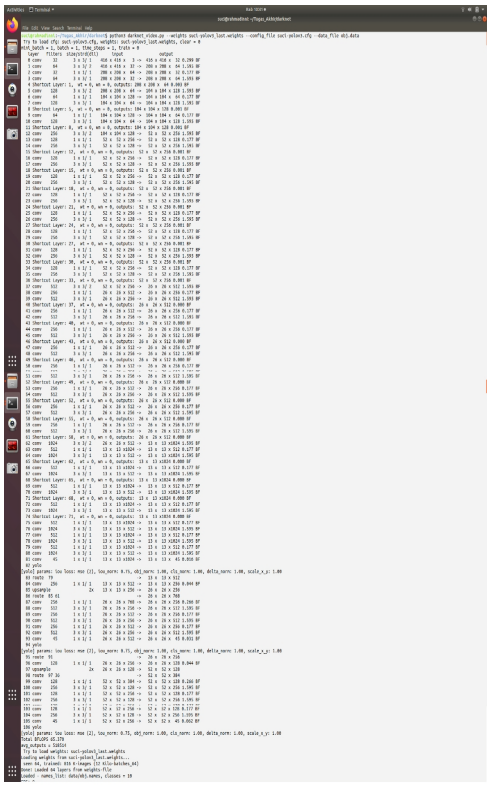
Gambar 9. Memulai Training

Setelah proses *training* selesai, maka *file* hasil *training* berupa sebuah file dengan ekstensi *.weights* di folder *darknet*/*backup* dan file tersebut digunakan pada proses *testing*.

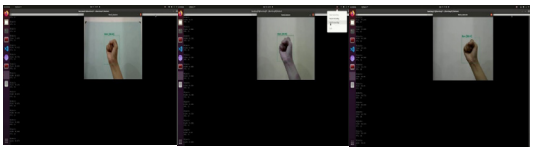
IV. Hasil dan Pembahasan

Pengujian data yang telah di *training* dilakukan dengan cara memanggil file *weights* hasil *training* ke dalam *command* di terminal *linux*. Proses *training* yang telah dilakukan dengan menggunakan *YOLO-v3* kemudian melakukan proses *testing* untuk melihat tingkat keberhasilan dari proses *training* yang telah dilakukan. *Command* untuk menjalankan proses *training* diperlihatkan pada Gambar 10. Terlihat bahwa *YOLO* -v3 mempunyai arsitektur 53 *layer* dan dipadatkan 53 *layer* lagi sehingga menjadi 106 *layer*. Proses pengujian data setelah proses *training* dilakukan dengan pengambilan sample video secara *real time*.

Pengujian dilakukan di sebuah ruangan dengan intensitas cahaya antara 15 sampai 35 *lux*. Berikut adalah hasil pengujian dari keberhasilan pengenalan dengan memasukkan data *weights* hasil *training*, *pass* menandakan *gesture* tangan terdeteksi dan *fail* menandakan tidak



Gambar 10. Memulai Testing



(a) (b) (c)

Gambar 11. Hasil deteksi gesture diam pada background (a) biru, (b) kuning, dan (c) putih

terdeteksi maupun ketidaksesuaian *gesture*. Adapun akurasi dan persentase keberhasilan dari tiap percobaan menggunakan perhitungan sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$Persentase\ Keberhasilan = \frac{Jumlah\ yang\ berhasil\ dideteksi}{Jumlah\ seluruh\ percobaan} \times 100\% \quad (6)$$

1. Pengujian terhadap *Background*

Pengujian terhadap *background* dilakukan dengan menggunakan 3 *background* yaitu biru, kuning, dan putih yang sebelumnya data dari *background* sudah diambil citranya dan di *training* terlebih dahulu. Pengujian dilakukan pada jarak 50 cm dengan intensitas cahaya sebesar 15 sampai 35 *lux* yang diukur dengan menggunakan *lux* meter.

Pada Gambar 11 menunjukkan hasil deteksi *gesture* tangan pada *background* biru, kuning, dan putih. Berdasarkan Gambar 11 sistem telah berhasil untuk mendeteksi *gesture* tangan dengan menggunakan 3 *background* yang berbeda.

Dari Tabel *confusion matrix*, dapat dilihat bahwa terdapat kesalahan deteksi beberapa *gesture* yaitu untuk *gesture* tiga, *yes*, dan *call*. Hal ini dikarenakan *gesture* tiga yang bentuknya hampir sama dengan *gesture* dua. Untuk *gesture yes* dan *call* terdapat kesalahan deteksi karena *gesture* tersebut mempunyai kemiripan.

Tabel 3 menunjukkan *confusion matrix background* putih, dari 10 kali percobaan disetiap *gesture*nya jumlah keseluruhan yang berhasil dideteksi dengan baik dan benar oleh sistem adalah 97 dan yang gagal adalah 3. Maka tingkat akurasinya menjadi:

Tabel 3. Confusion matrix Background Putih

Pr / Ak	1	2	3	4	hello	yes	no	ok	call	diam
1	10	0	0	0	0	0	0	0	0	0
2	0	10	0	0	0	0	0	0	0	0
3	0	1	9	0	0	0	0	0	0	0
4	0	0	0	10	0	0	0	0	0	0
hello	0	0	0	0	10	0	0	0	0	0
yes	0	0	0	0	0	9	0	0	1	0
no	0	0	0	0	0	0	10	0	0	0
ok	0	0	0	0	0	0	0	10	0	0
call	0	0	0	0	0	1	0	0	9	0
diam	0	0	0	0	0	0	0	0	0	10

Tabel 4. Confusion matrix Background Kuning

Pr / Ak	1	2	3	4	hello	yes	no	ok	call	diam
1	10	0	0	0	0	0	0	0	0	0
2	0	10	0	0	0	0	0	0	0	0
3	0	1	9	0	0	0	0	0	0	0
4	0	0	0	10	0	0	0	0	0	0
hello	0	0	0	0	10	0	0	0	0	0
yes	0	0	0	0	0	10	0	0	0	0
no	0	0	0	0	0	0	10	0	0	0
ok	0	0	0	0	0	0	0	10	0	0
call	0	0	0	0	0	1	0	0	9	0
diam	0	0	0	0	0	0	0	0	0	10

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Akurasi = \frac{97 + 0}{97 + 0 + 0 + 3} = 0{,}97 \times 100\%$$

Pada Tabel 4 *confusion matrix background* kuning, dari 10 kali percobaan disetiap *gesture*nya jumlah keseluruhan yang berhasil dideteksi dengan baik dan benar oleh sistem adalah 98 dan yang gagal adalah 2. Maka persentase keberhasilannya menjadi:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Akurasi = \frac{98 + 0}{98 + 0 + 0 + 2} = 0{,}98 \times 100\% = 98\%$$

Pada Tabel 5 *confusion matrix background* biru, dari 10 kali percobaan disetiap *gesture*nya jumlah keseluruhan yang berhasil dideteksi dengan baik dan benar oleh sistem adalah 98 dan yang gagal adalah 2. Maka akurasinya menjadi:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

Tabel 5. Confusion matrix Background Biru

Pr / Ak	1	2	3	4	hello	yes	no	ok	call	diam
1	10	0	0	0	0	0	0	0	0	0
2	0	10	0	0	0	0	0	0	0	0
3	0	0	10	0	0	0	0	0	0	0
4	0	0	0	10	0	0	0	0	0	0
hello	0	0	0	0	10	0	0	0	0	0
yes	0	0	0	0	0	9	0	0	1	0
no	0	0	0	0	0	0	10	0	0	0
ok	0	0	0	0	0	0	0	10	0	0
call	0	0	0	0	0	1	0	0	9	0
diam	0	0	0	0	0	0	0	0	0	10

Tabel 6. Pengujian Terhadap Intensitas Cahaya

Isyarat	20%		50%		80%		100%	
	Pass	Fail	Pass	Fail	Pass	Fail	Pass	Fail
1	10	0	10	0	10	0	10	0
2	10	0	10	0	10	0	10	0
3	10	0	10	0	9	1	10	0
4	10	0	10	0	10	0	10	0
Hi	10	0	10	0	10	0	10	0
Yes	10	0	9	1	10	0	9	1
No	10	0	10	0	10	0	10	0
Ok	10	0	10	0	10	0	10	0
Call	10	0	10	0	10	0	9	1
Diam	10	0	10	0	10	0	10	0
Total	100	0	99	1	99	1	98	0
Rata – rata Error	0		0,01		0,01		0,02	
Persentase Keberhasilan	100%		99%		99%		98%	

$$Akurasi = \frac{98+0}{98+0+0+2} = 0,98 \times 100\% = 98\%$$

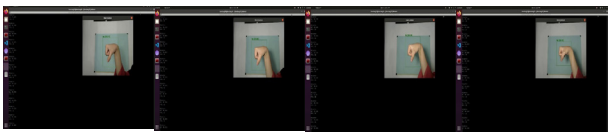
Berdasarkan pengujian yang telah dilakukan dari keseluruhan percobaan dapat disimpulkan bahwa sistem dapat mendeteksi *gesture* tangan meskipun *gesture* tangan berada pada *background* yang berbeda – beda.

Hal ini dikarenakan data sebelum *training* menggunakan *background* yang sama dengan *background* pada pengujian. Dengan kata lain apabila ingin mendeteksi dengan *background* yang lain maka data *background*nya harus di *training* terlebih dahulu. Dan ada beberapa *gesture* yang mengalami *fail* / gagal mendeteksi saat pengujian, yaitu *gesture* tiga, *yes,* dan *call.* Hal ini dikarenakan *gesture* tiga yang bentuknya hampir sama dengan *gesture* dua, serta *gesture yes* dan *call* karena *gesture* tersebut mempunyai kemiripan. Untuk itu, ketepatan dalam pengambilan data untuk proses *training* mempengaruhi hasil deteksi untuk keberhasilan sistem.

2. Pengujian Terhadap Intensitas Cahaya

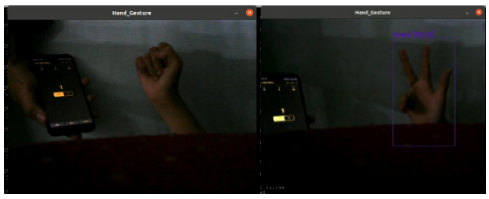
Pengujian terhadap intensitas cahaya dilakukan dengan menggunakan bantuan *dimmer switch* dan sebuah lampu sehingga cahaya dari lampu dapat disesuaikan dengan rotasi pada *dimmer switch*. Cahaya diatur dengan persentase 20%, 50%, 80%, dan 100% pada *dimmer switch*. Cahaya 20% artinya dimmer diputar sampai dengan 20% dengan intensitas cahaya antara 90 – 200 *lux*. Cahaya 50% artinya dimmer diputar sampai dengan 50% dengan intensitas cahaya antara 800 – 1050 *lux*. Cahaya 80% artinya dimmer diputar sampai dengan 80% dengan intensitas cahaya 1900 – 2500 *lux*. Cahaya 100% artinya dimmer diputar sampai dengan 100% dengan intensitas cahaya antara 2700 – 3100 *lux*.

Pada Gambar 12 menampilkan hasil pengujian bahwa sistem telah berhasil untuk mendeteksi *gesture* tangan dengan intensitas cahaya yang berbeda – beda.



(a) (b) (c) (d)

Gambar 12. Hasil deteksi gesture No pada Intensitas cahaya (a) 20%, (b) 50%, (c) 80%, (d) 100%.



Gambar 13. Contoh Gesture yang tidak terdeteksi dan kesalahan deteksi

Berdasarkan pengujian yang telah dilakukan dari keseluruhan percobaan dapat disimpulkan bahwa sistem dapat mendeteksi *gesture* tangan meskipun intensitas cahaya saat pendeteksian dinaikkan. Meskipun demikian persentase keberhasilan sistem dipengaruhi oleh data yang diambil untuk proses *training* sehingga tingkat keberhasilannya tidak sampai 100%. Saat pengujian sistem juga dicoba dengan kondisi intensitas cahaya yang rendah, hasilnya sistem tidak dapat mendeteksi *gesture* tangan saat ruangan berada di kondisi intensitas cahaya yang rendah dengan intensitas cahaya 0 – 5 *lux*.

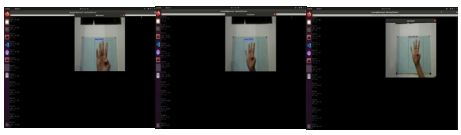
Pada Gambar 13 menunjukkan hasil deteksi *gesture* tangan dengan intensitas cahaya 0 sampai 5 *lux*. Dapat dilihat bahwa terjadi kesalahan pada sistem saat mendeteksi *gesture* yang diberikan. Hal ini dikarenakan intensitas cahaya yang sangat rendah sehingga sistem tidak dapat mengenali maupun mendeteksi *gesture* tangan yang diberikan.

3. Pengujian Terhadap Jarak

Pengujian terhadap jarak dilakukan dengan jarak 30 cm, 50 cm, dan 80 cm dari objek terhadap kamera. Pengujian dilakukan dengan intensitas cahaya dan *background* yang sama.

Pada Gambar 14 menampilkan hasil pengujian bahwa sistem telah berhasil untuk mendeteksi *gesture* tangan dengan jarak 30 cm, 50 cm, dan 80 cm. Pada Tabel 7 menampilkan hasil deteksi pengujian sistem pada jarak 30 cm, 50 cm, dan 80 cm setelah dilakukan percobaan sebanyak 10 kali.

Dari seluruh pengujian *gesture* tangan terhadap jarak, hasil pengenalan *gesture* tangan dengan jarak 50 cm lebih



(a) (b) (c)

Gambar 14. Hasil deteksi gesture Empat pada Jarak (a) 30 cm, (b) 50 cm, dan (c) 80 cm.

Tabel 7. Confusion matrix Jarak 30 cm

Pr / Ak	1	2	3	4	hello	yes	no	ok	call	diam
1	10	0	0	0	0	0	0	0	0	0
2	0	10	0	0	0	0	0	0	0	0
3	0	1	9	0	0	0	0	0	0	0
4	0	0	0	10	0	0	0	0	0	0
hello	0	0	0	0	10	0	0	0	0	0
yes	0	0	0	0	0	10	0	0	0	0
no	0	0	0	0	0	0	10	0	0	0
ok	0	0	0	0	0	0	0	10	0	0
call	0	0	0	0	0	0	0	0	10	0
diam	0	0	0	0	0	0	0	0	0	10

Tabel 9. Confusion matrix Jarak 80 cm

Pr / Ak	1	2	3	4	hello	yes	no	ok	call	diam
1	5	5	0	0	0	0	0	0	0	0
2	2	5	3	0	0	0	0	0	0	0
3	2	3	3	2	0	0	0	0	0	0
4	0	0	3	4	3	0	0	0	0	0
hello	0	0	0	4	6	0	0	0	0	0
yes	0	0	0	0	0	3	2	0	3	2
no	0	0	0	0	0	3	4	0	3	0
ok	0	0	0	0	0	3	0	4	3	0
call	0	0	0	0	0	4	2	1	3	0
diam	0	0	0	0	0	2	2	0	0	6

akurat dibandingkan dengan jarak 30 cm maupun 80 cm. Tabel 8 dan Tabel 9 menampilkan dan membandingkan nilai aktual dari nilai prediksi pada pengujian *gesture* tangan terhadap jarak. Jarak 30 cm dan 50 cm lebih akurat dari pada jarak 80 cm. Hal ini dikarenakan pada saat pengambilan data yang diambil pada jarak rentang 30 cm dan 80 cm sehingga untuk data pada jarak 30 cm dan 50 cm lebih bagus serta untuk jarak 80 cm datanya kurang bagus sehingga hasil akurasi pada jarak 80 cm kurang dari 0,5.

Berikut tingkat akurasi untuk tiap masing-masing jarak.

Jarak 30 cm:

$$Akurasi = \frac{99+0}{99+0+0+1} = 0,99 \times 100\% = 99\%$$

Jarak 50 cm:

$$Akurasi = \frac{100+0}{100+0+0+0} = 1 \times 100\% = 100\%$$

Jarak 80 cm:

$$Akurasi = \frac{43+0}{43+0+0+57} = 0,43 \times 100\% = 43\%$$

Berdasarkan pengujian yang telah dilakukan dapat diambil kesimpulan bahwa ketepatan pengambilan

Tabel 8. Confusion matrix Jarak 50 cm

Pr / Ak	1	2	3	4	hello	yes	no	ok	call	diam
1	10	0	0	0	0	0	0	0	0	0
2	0	10	0	0	0	0	0	0	0	0
3	0	0	10	0	0	0	0	0	0	0
4	0	0	0	10	0	0	0	0	0	0
hello	0	0	0	0	10	0	0	0	0	0
yes	0	0	0	0	0	10	0	0	0	0
no	0	0	0	0	0	0	10	0	0	0
ok	0	0	0	0	0	0	0	10	0	0
call	0	0	0	0	0	0	0	0	10	0
diam	0	0	0	0	0	0	0	0	0	10

data pada untuk proses *training* berpengaruh terhadap keberhasilan sistem. Pada perhitungan diatas dapat dilihat bahwa persentase keberhasilan untuk jarak 30 cm adalah 99% dan untuk jarak 50 cm adalah 100%. Persentase keberhasilan ini lebih tinggi daripada persentase keberhasilan untuk jarak 80 cm yang hanya 43%. Namun, ini lebih baik dari pada metode sebelumnya yang menggunakan metode segmentasi warna dan *neural network backpropagation* [5] yang hanya dapat mendeteksi pada jarak 30 cm dan 35 cm dengan rata – rata persentase keberhasilan 87%. Untuk itu agar sistem dapat mendeteksi lebih akurat maka pengambilan dataset sangat mempengaruhi untuk hasil akhir dari pendeteksian.

V. KESIMPULAN

Dari pengujian yang telah dilakukan, sistem pendeteksian *gesture* tangan dengan menggunakan *image processing* yaitu *YOLO-v3* mampu mendeteksi dan mengklasifikasi *gesture* tangan. Persentase keberhasilan dari sistem untuk mendeteksi dan mengklasifikasi *gesture* tangan dengan *background* putih sebesar 97%, *background* biru sebesar 98%, dan *background* kuning 98%. Sistem pendeteksian ini mampu mendeteksi dan mengklasifikasi *gesture* tangan meskipun intensitas cahaya berubah – ubah. Persentase keberhasilan pendeteksian untuk intensitas cahaya 20 % sebesar 100 %, intensitas cahaya 50 % sebesar 99%, intensitas cahaya 80 % sebesar 99 %, dan intensitas cahaya 100 % sebesar 98%. Sistem dapat mendeteksi dan mengklasifikasi *gesture* tangan dengan baik pada saat intensitas cahaya antara 15 *lux* sampai 3100 *lux*. Saat intensitas cahaya yang sangat rendah yaitu sekitar 0 sampai 5 *lux* sistem tidak dapat mengenali maupun mengklasifikasi *gesture* tangan yang diberikan. Sistem pendeteksian ini mampu mendeteksi dan mengklasifikasi *gesture* tangan pada rentang jarak 30 cm sampai 80 cm. Saat jarak pendeteksian 80 cm sistem masih dapat mendeteksi, namun hasil deteksinya kurang akurat bahkan tidak terdeteksi sama sekali. Dari semua pengujian yang telah dilakukan sistem dapat mendeteksi dan mengklasifikasi *gesture* tangan dengan persentase keberhasilan lebih dari 90%. Hal ini dipengaruhi oleh

bentuk dari *gesture* tangan yang hampir mirip, sehingga ketepatan pengambilan *dataset* sebelum *training* dan banyaknya data yang di *training* sangat mempengaruhi untuk proses pengujian. Semakin banyak *dataset* dan semakin bagus *dataset* yang di *training* maka hasil deteksi akan semakin baik.

Referensi

[1] I. Steinberg, T. M. London and C. D. Dotan, "Hand Gesture Recognition in Images dan and Video," CCIT Report, vol. 763, pp. 1-20, 2010.

[2] W. Gazali and H. Soeparno, "Pendeteksian Gerak Tangan Manusia Sebagai Input Pada Komputer," Mat Sat, vol. 11, no. 2, pp. 129-137, 2011.

[3] M. D. Rosyadi, F. Hafidh and M. Y. Kurniawan, "Pengenalan Real Time Abjad Bahasa Isyarat Indonesia Menggunakan Segmentasi YCbCr," vol. 2, no. 2, pp. 1-5, 2017.

[4] F. Asriani and H. Susilawati, "Pengenalan Isyarat Tangan Statis Pada Sistem Isyarat Bahasa Indonesia Berbasis Jaringan Syaraf Tiruan Perambatan Balik," Makara Journal of Technology, vol. 14, no. 2, pp. 150-154, 2010.

[5] S. E. Octaviani, "Tracking dan Pengenalan Pola Gerak Tangan Berbasis Penginderaan Visual," Tugas Akhir, Politeknik Negeri Batam, Batam, 2019.

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[7] L. Chen, F. Wang, H. Deng and K. Ji, "A Survey on Hand Gesture Recognition," Conference: Proceedings of the 2013 International Conference on Computer Sciences and Applications, vol. DOI:10.1109/CSA.2013.79, 2013.

[8] Kanchan Dabre and D. Surekha, "Machine Learning Model for Sign Language Interpretation using Webcam Images," *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pp. 317-321, 2014.

[9] L. Pigou, S. Dieleman and P.-J. Kindermas, "Sign Language Recognition Using Convolutional Neural Networks," pp. 572-578, 2015

[10] A. Y. R, "YOLO (you only look once)", *Universitas Gadjah Mada Menara Ilmu Machine Learning*, 5 Agustus 2018. [Online]. Available: https://machinelearning.mipa.ugm.ac.id/2018/08/05/yolo-you-only-look-once/. [Accessed 20 Apr 2021].

[11] D. Karunakaran, "Deep learning series 1: Intro to deep learning", *Medium*, 23 April 2018. [Online]. Available: https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20. [Accessed 20 Apr 2021].

[12] J. W. G. Putra, Pengenalan Konsep Pembelajaran Mesin dan Deep Learning, self-published work, 2020.

[13] S. Sena, "Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN)", *Medium*, 13 Nov 2017. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94. [Accessed 20 Apr 2021].

[14] Fawwaz, M. A. A. Fawwaz, K. N. Ramadhani and F. Sthevanie, "Klasifikasi Ras pada Kucing menggunakan Algoritma Convolutional Neural Network(CNN)," e-Proceeding of Engineering, vol. 8, no. 1, pp. 715-730, 2021.

[15] Hanin, M. A. Hanin, R. Patmasari and R. Y. N. Fu'adah, "Sistem Klasifikasi Penyakit Kulit Menggunakan Convolutional Neural Network (CNN)," e-Proceeding of Engineering, vol. 8, no. 1, pp. 273-281, 2021.

[16] S. E. Rudiawan, R. Analia, D. S. P and H. Soebakti, "The Deep Learning Development for Real-Time Ball and Goal Detection of Barelang-FC," *International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, September 2017.

[17] Q. Aini, N. Lutfiani, H. Kusumah and M. S. Zahran, "Deteksi dan Pengenalan Objek Dengan Model Machine Learning: Model YOLO," CESS (Journal of Computer Engineering System and Science), vol. 6, no. 2, pp. 192-199, 2021.

[18] M. Mantripragada, "Digging deep into YOLO V3 - A hands-on guide Part 1", *Towards Data Science*, 16 August 2020. [Online]. Available: https://towardsdatascience.com/digging-deep-into-YOLO-v3-a-hands-on-guide-part-1-78681f2c7e29. [Accessed January 2021].

[19] B. P. G. Pamungkas, B. Nugroho and F. Anggraeny, "Deteksi dan Menghitung Manusia Menggunakan YOLO_CNN," Jurnal Informatika dan Sistem Informasi (JIFoSI), vol. 02, no. 1, pp. 67-76, 2021.

[20] F. Indaryanto, A. Nugroho and A. F. Suni, "Aplikasi Penghitung Jarak dan Jumlah Orang Berbasis YOLO Sebagai Protokol Kesehatan Covid-19," Edu Kompatika, vol. 8, no. 1, pp. 31-38, 2021.