Institut Riset dan Publikasi Indonesia (IRPI)
MALCOM: Indonesian Journal of Machine Learning and Computer Science
Journal Homepage: https://journal.irpi.or.id/index.php/malcom
Vol. 5 Iss. 3 July 2025, pp: 1061-1073
ISSN(P): 2797-2313 | ISSN(E): 2775-8575

YOLO11 and OpenCV Implementation for Phrase Recognition in Real-Time Hand Sign Language Videos

Implementasi YOLO11 dan OpenCV untuk Pengenalan Frasa dalam Video *Real-Time* Bahasa Isyarat Tangan

Henoch Yanuar Ari Swasono[1*], Agustinus Rudatyo Himamunanto[2], Febe Maedjaja[3]

[1,2,3]Program Studi Informatika, Fakultas Sains dan Komputer
Universitas Kristen Immanuel, Indonesia

E-Mail: [1]henoch.yanuar.a@mail.ukrim.ac.id, [2]rudatyo@ukrimuniversity.ac.id,
[3]febe@ukrimuniversity.ac.id

Received Jun 23th 2025; Revised Jul 23th 2025; Accepted Jul 30th 2025; Available Online Jul 31th 2025, Published Aug 15th 2025
Corresponding Author: Henoch Yanuar Ari Swasono

Abstract

Sign language is the primary means of communication for individuals with hearing and speech impairments. However, the limited public understanding of sign language often becomes a barrier in effective communication. This study aims to design a real-time recognition program for Bahasa Isyarat Indonesia (BISINDO) hand sign phrases using the YOLO11 algorithm and the OpenCV library. YOLO11 is used as a deep learning method to recognize hand gestures, while OpenCV is utilized for real-time video processing and visualization of detection results. The model was trained using more than 3,000 images representing six BISINDO phrase classes: "saya", "kamu", "senang", "bingung", "marah", and "apa kabar", for 263 epochs. Model testing results showed average precision and recall values above 0.9, an F1-Score of 0.982, mAP50 of 0.993, and mAP50-95 of 0.938. In real-time testing, the model demonstrated stable average latency in the range of 80–90ms, a frame rate of 11–12 FPS, and an average confidence score of 0.9 across all classes. Based on the research that has been done, it is concluded that the integration of YOLO11 and OpenCV is successfully used as an algorithm in recognizing BISINDO hand sign language phrases in real-time.

Keyword: BISINDO, Deep Learning, OpenCV, YOLO11

Abstrak

Bahasa isyarat adalah alat komunikasi utama bagi para penyandang tunarungu dan tunawicara. Namun, terbatasnya pemahaman bahasa isyarat oleh masyarakat umum sering kali menjadi kendala dalam berkomunikasi. Penelitian ini bertujuan untuk merancang program pengenalan frasa bahasa isyarat tangan Bahasa Isyarat Indonesia (BISINDO) secara *real-time* dengan menggunakan algoritma YOLO11 dan library OpenCV. YOLO11 digunakan sebagai metode *deep learning* untuk mengenali isyarat tangan, sedangkan OpenCV digunakan untuk pemrosesan video *real-time* dan visualisasi hasil deteksi. Model ini dilatih menggunakan lebih dari 3.000 gambar yang mewakili enam *class* frasa BISINDO: “saya”, “kamu”, “senang”, “bingung”, “marah”, dan “apa kabar”, sebanyak 263 *epoch*. Hasil pengujian model menunjukkan rata-rata nilai *precision* dan *recall* di atas 0,9; *F1-Score* sebesar 0,982; *mAP50* sebesar 0,993; dan *mAP50-95* sebesar 0,938. Pada pengujian *real-time*, model menunjukkan *latency* rata-rata stabil di kisaran 80-90ms, *frame rate* 11-12FPS, dan *confidence score* rata-rata 0,9 untuk semua *class*. Berdasarkan Penelitian yang telah dilakukan, disimpulkan bahwa integrasi YOLO11 dan OpenCV berhasil digunakan sebagai algoritma dalam mengenali frasa bahasa isyarat tangan BISINDO secara *real-time*.

Kata Kunci: BISINDO, *Deep Learning*, OpenCV, YOLO11

1. PENDAHULUAN

Komunikasi adalah aspek fundamental dalam kehidupan manusia. Bagi orang dengan gangguan pendengaran atau gangguan bicara, bahasa isyarat seperti Bahasa Isyarat Indonesia (BISINDO), merupakan alat komunikasi utama. Sensus penduduk yang dilakukan pada tahun 2020 menunjukkan bahwa 1,43% dari populasi Indonesia adalah orang dengan disabilitas, di mana 0,36% di antaranya memiliki gangguan

pendengaran dan 0,35% memiliki gangguan bicara [1]. Namun, interaksi meraka masih terbatas karena tidak semua orang dapat memahami bahasa isyarat [2]. Kondisi ini seringkali menyebabkan hambatan dalam pendidikan, pekerjaan, dan layanan publik. Saat ini, sebagian besar teknologi pengenalan bahasa isyarat masih berfokus pada *American Sign Language* (ASL), sementara penelitian dan implementasi untuk BISINDO masih sangat terbatas, sehingga belum cukup memfasilitasi kebutuhan di Indonesia [1], [3]. Oleh karena itu, pada penelitian ini penulis akan bereksperimen membuat sebuah program yang dapat digunakan untuk mengenali enam frasa bahasa isyarat BISINDO secara *real-time*.

Tantangan utama dalam mengenali bahasa isyarat adalah kompleksitas elemen-elemennya. Gerakan tangan yang cepat dan perubahan posisi yang dinamis , serta variasi regional dan budaya membuat pengenalan bahasa isyarat menjadi tugas yang sangat rumit. Selain itu, setiap bahasa isyarat memiliki variasi regional dan budaya, sehingga semakin sulit untuk mengembangkan program pengenalan yang akurat dan andal.

You Only Look Once (YOLO) adalah sebuah metode yang digunakan untuk mendeteksi dan mengenali objek dengan menawarkan keakuratan dan kecepatan operasinya. Metode ini bergantung pada jaringan Convolution Neural Network (CNN) [4]. YOLO dapat memberikan hasil yang lebih akurat dan memiliki probabilitas tertinggi dibandingkan dengan R-CNN dan Faster R-CNN, sehingga YOLO dapat mendeteksi objek dengan efisien tanpa mengurangi performa kinerjanya [5]. Penelitian sebelumnya juga menyatakan bahwa seri YOLO memiliki waktu inferensi tercepat dibandingkan dengan Faster R-CNN. YOLO11 adalah model tercepat dengan waktu inferensi 13,5 milidetik (*ms*), YOLO8 dan YOLO10 dengan waktu inferensi berturut-turut 23 dan 19,3ms. Sedangkan Faster R-CNN memiliki waktu inferensi lebih lama sebesar 63,8ms. Hasil ini menunjukkan bahwa algoritma dari seri YOLO memiliki potensi untuk digunakan secara *real-time* [6].

YOLO11 mewakili kemajuan signifikan dalam *computer vision*. Versi 11 dari YOLO ini menunjukkan peningkatan yang signifikan dalam akurasi, kecepatan pemrosesan, fleksibilitas, serta mengurangi jumlah parameter yang diperlukan [7]. Dari peningkatan ini, YOLO11 sangat cocok digunakan untuk pengenalan bahasa isyarat secara *real-time*. Selain itu, Open Computer Vision (OpenCV) juga dapat digunakan untuk mendukung deteksi objek secara *real-time*. OpenCV adalah pustaka sumber terbuka yang dirancang untuk pemrosesan gambar. OpenCV dapat membuat komputer memiliki kemampuan "melihat" yang serupa dengan pemrosesan visual manusia. OpenCV menyediakan banyak algoritma dasar dan modul *object detection* untuk *computer vision* [8].

Penelitian deteksi bahasa isyarat tangan pernah dilakukan oleh Arifah et al. (2022) menggunakan metode YOLO dan CNN untuk video percakapan yang bertujuan utnuk identifikasi dan klasifikasi gerakan tangan. Penelitian ini menghasilkan nilai akurasi sebesar 89% [9]. Penelitian serupa juga dilakukan oleh S. Daniels (2020) yang bertujuan untuk pengembangan sistem pengenalan bahasa isyarat huruf yang dapat membaca masukan dari data video secara *real-time* menggunakan metode CNN dengan arsitektur YOLO [10]. Penelitian oleh Mujahid et al. (2021) menghasilkan model ringan berdasarkan YOLOv3 dan jaringan saraf *convolutional* DarkNet-53 yang dapat digunakan untuk deteksi *real-time*, baik untuk gambar tangan statis maupun dinamis yang berupa video [11]. Penelitian oleh Nur'azizan et al. (2024) yang menggunakan metode OpenCV dan MediaPipe mendapatkan hasil akurasi model dengan nilai *F1-Score*, *recall*, dan *precision* berturut-turut sebesar 0,9875; 0,9875; dan 0,9875 untuk pengenalan bahasa isyarat secara *real-time* [12]. Penelitian oleh Nurul Renaningtias et al. (2025) yang menggunakan metode YOLOv7 dan OpenCV untuk deteksi alfabet BISINDO secara *real-time*, menghasilkan nilai *mAP@IoU* 0,5 sebesar 0,995; *recall* 1,0; *precision* 1,0; *F1-Score* 1,0. Namun, performa model pada pengujian secara *real-time* tidak sebaik hasil saat pelatihan [13].

Berbeda dengan penelitian-penelitian sebelumnya yang lebih berfokus pada pengenalan isyarat tangan secara individu atau pengenalan alfabet BISINDO, penelitian ini mengusung pendekatan baru dengan mengintegrasikan YOLO versi 11 (YOLO11) dan OpenCV untuk pengenalan frasa dalam bahasa isyarat tangan BISINDO pada video *real-time*. Penelitian Arifah et al. (2022) dan Daniels (2020) menitikberatkan deteksi isyarat tangan menggunakan YOLO dan CNN untuk mengenali bahasa isyarat tangan, namun hanya sebatas pengenalan isyarat angka dan alfabet bukan frasa. Sementara itu, Mujahid et al. (2021) berfokus pada pengenalan gerakan tangan dengan YOLOv3 tanpa pra pemrosesan tambahan, namun hanya meneliti pengenalan angka bukan frasa. Penelitian Nur'azizan et al. (2024) menggunakan OpenCV dan MediaPipe untuk deteksi bahasa isyarat secara *real-time*, tetapi tidak mengeksplorasi metode YOLO dalam penelitian. Sedangkan penelitian Renaningtias et al. (2025) menggunakan YOLOv7 dan OpenCV untuk pendeteksian BISINDO, namun tidak menggunakan YOLO11 dan hanya mendeteksi alfabet, bukan frasa. Sejumlah keterbatasan tersebut menjadi celah yang akan diisi oleh penelitian ini. Dengan demikian, kebaruan penelitian ini terletak pada kombinasi metode YOLO11 dan OpenCV untuk pengenalan frasa BISINDO secara *real-time*, yang belum pernah diterapkan dalam studi sebelumnya.

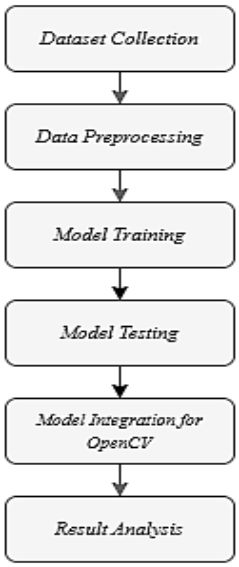
Penelitian ini dilakukan untuk menguji apakah metode YOLO11 yang diintegrasikan dengan *library* OpenCV dapat mengenali frasa dalam bahasa isyarat tangan BISINDO. Penggabungan kedua teknologi tersebut, diharapkan dapat dihasilkan sebuah program yang memiliki akurasi tinggi dan mampu memberikan

kontribusi dalam pengenalan bahasa isyarat tangan secara *real-time*, serta menjadi landasan untuk pengembangan sistem yang lebih maju pada masa mendatang. Dengan demikian, penelitian ini tidak hanya memiliki nilai akademis untuk mengembangkan metode yang lebih maju dalam pengenalan bahasa isyarat, tetapi juga memiliki nilai praktis, terutama dalam membantu seseorang yang menggunakan bahasa isyarat untuk berkomunikasi dengan orang-orang di sekitarnya.

2. METODE PENELITIAN

Penelitian ini menggunakan metode dengan pendekatan *deep learning* yaitu YOLO11. YOLO11 memiliki keunggulan pada penerapan *object detection* secara *real-time* [14]. Hal ini membuat penggunaan YOLO11 sangat penting karena bahasa isyarat tangan adalah bentuk dari komunikasi non-verbal manusia yang mencakup berbagai bentuk gerakan dinamis, posisi jari, atau bentuk tangan yang spesifik [15], sehingga pendeteksian objek tangan dapat lebih akurat dan dapat menciptakan *user experience* yang baik.

Penelitian ini memiliki 6 tahapan proses yang harus dilakukan seperti pada Gambar 1, yaitu: *Dataset Collection*, *Data Preprocessing*, *Model Training*, *Model Testing*, *Model Integration for OpenCV*, dan *Result Analysis*.



Gambar 1. Tahapan penelitian

2.1. *Dataset Collection*

Salah satu tahapan yang paling penting dari *machine learning* adalah pengumpulan *dataset*. Hal ini penting karena kualitas model hasil pelatihan bergantung pada seberapa baik kualitas *dataset* yang digunakan juga [16]. Oleh karena itu, *dataset* yang digunakan harus heterogen karena terdapat beberapa frasa yang digunakan pada penelitian ini [17].

Dataset yang digunakan pada penelitian ini dikumpulkan dari *Kaggle* dan hasil pengambilan data secara manual melalui *webcam*. *Dataset* yang digunakan pada penelitian ini berupa gambar yang terdiri dari enam frasa bahasa isyarat tangan BISINDO, yaitu: “saya”, “kamu”, “senang”, “bingung”, “marah”, dan “apa kabar”. *Dataset* dibagi menjadi dua dengan perbandingan 80:20, 80% sebagai data pelatihan *(train)* dan 20% sebagai data validasi *(val)* [10], yang menghasilkan 2.817 data *train* dan 704 data *val*.

Selain itu, disediakan juga data primer sebagai data pengujian (*test)* yang diambil dari empat partisipan sebagai subjek penelitian yang tidak dilibatkan dalam proses pelatihan. Setiap *class* terdapat 30 data *test* dari satu partisipan, sehingga terdapat 120 data *test* untuk setiap *class* atau 720 data *test* dari keseluruhan *class*. Persebaran jumlah data per *class* disajikan pada Tabel 1.

Tabel 1. Persebaran Jumlah Data pada Dataset

Class	Train	Val	Test
Saya	339	85	120
Kamu	444	111	120
Senang	466	117	120
Bingung	488	122	120
Marah	526	131	120
Apa Kabar	554	138	120

2.2. *Data Preprocessing*

Preprocessing dilakukan sebelum data digunakan untuk pelatihan model [11]. *Dataset* yang sudah dikumpulkan diproses terlebih dahulu dengan tujuan membuat data yang tidak terstruktur menjadi lebih terstruktur dengan bentuk standar, memastikan kualitas *dataset* sebagai input, serta mengekstrak informasi dari *dataset* [14][3]. Beberapa langkah pemrosesan dilakukan pada penelitian ini terganatung dengan jenis data, sebagai berikut:

1. *Framing*
   Sebagian *dataset* yang dikumpulkan dari sumber terbuka berbentuk video, karena bahasa isyarat yang digunakan merupakan gerakan tangan dinamis. Hal ini membuat *dataset* video perlu malalui tahap *framing* agar data berubah menjadi gambar yang berurut, sehingga dapat digunakan untuk pelatihan model. Proses ini melibatkan penggunaan OpenCV sebagai alat untuk pemrosesan video [8].

2. *Augmentation*
   Augmentation data adalah tahapan yang sangat penting dalam *object detection*. Proses ini terdiri dari beberapa teknik yang mengubah ukuran dan kualitas kumpulan data pelatihan [14], serta melakukan penggandaan data gambar yang sudah ada. *Augmentation* dilakukan dengan menggunakan OpenCV yang dapat digunakan untuk mentransformasi gambar [4], dimana OpenCV akan membaca data gambar yang berupa piksel untuk mentrasformasikan gambar tersebut, dan menyimpan hasil sebagai file baru. Transformasi gambar dilakukan dengan cara pembalikan secara horizontal, meningkatkan kecerahan, dan mengubah warna ke skala abu-abu *(grayscale)*. Dengan proses ini, satau gambar dapat digandakan menjadi empat gambar sehingga menambah variasi data gambar agar model tetap dapat mendeteksi objek dalam berbagai kondisi pencahayaan dan kualitas kamera [7][9]. Dengan menggunakan teknik *augmentation*, masalah keterbatasan data yang dapat menyebabkan model *overfitting* dapat teratasi [14] [18].

3. *Labeling*
   Semua data yang digunakan untuk melatih model harus diberi label. Pelabelan data berarti memberikan nama *class* serta anotasi pada gambar [19]. Pelabelan dilakukan menggunakan alat yang bernama *LabelImg*, ini adalah alat yang khusus digunakan untuk pelabelan. Pelabelan harus dilakukan agar data dapat digunakan untuk pelatihan model sehingga YOLO dapat mengenali objek pada setiap gambar [17].

2.3. *Model Training*

Tahap ini adalah tahapan paling utama pada penelitian ini yang menggunakan algoritma YOLO11 sebagai metodenya. YOLO11 memiliki arsitektur yang lebih canggih daripada versi-versi sebelumnya. YOLO11 dengan memperkenalkan konsep utama YOLOv10, menggunakan *anchor-free* berbasis *decoupled head*, di mana *regression branch* menggunakan konvolusi normal dan *classification head* menggunakan konvolusi yang dapat dipisahkan secara kedalaman *(DWConv)*. sehingga dapat mengurangi perhitungan yang berlebihan secara efektif [20]. Selain itu, YOLO11 juga memperkenalkan modul *Cross-Stage Partial with Self-Attention (*C2PSA*)*. Modul C2PSA menggabungkan keunggulan jaringan *cross-stage* partial dengan mekanisme *self-attention*. Modul C2PSA memastikan model dapat menangkap informasi kontekstual secara lebih efektif dan meningkatkan akurasi deteksi objek [21].Oleh karena itu, algoritma ini dapat menghasilkan model dengan *precision* dan *performance* yang lebih seimbang [6]. Pada tahap ini, semua data *train* yang telah melalui tahap *Preprocessing Data* akan digunakan untuk pelatihan model.

Pelatihan model dilakukan dengan teknik *transfer learning*. *Transfer learning* merupakan sebuah teknik penggunaan model yang sudah dilatih sebelumnya *(pre-trained model)* atau model yang sudah disediakan oleh YOLO sendiri untuk melatih dengan *dataset* baru tanpa harus melakukan pelatihan model dari awal [22]. Teknik ini mempu menghasilkan model baru dengan performa lebih tinggi serta waktu pelatihan yang lebih cepat dengan data yang lebih sedikit, karena pelatihan menggunakan *knowlage* dari model sebelumnya. Dengan demikian, ini memberikan keuntungan yang lebih besar daripada melatih model dari awal, serta proses pelatihan model menjadi lebih efisien [23]. Selama pelatiahn, YOLO akan me-*resize* semua data gambar. Gambar akan diperkecil secara horizontal maupun vertikal menjadi $320\text{px} \times 320\text{px}$. Hal ini bertujuan untuk membuat agar proses pelatihan model lebih ringan dan cepat [22], serta model yang dihasilkan lebih ringan juga.

2.4. *Model Testing*

Model yang dihasilkan dari pelatihan dengan *transfer learning* pada tahap sebelumnya kemudian diuji terlebih dahulu. Pengujian dilakukan dengan data *test* yang sudah disiapkan dalam *dataset* untuk menganalisis performa model. Beberapa parameter yang dianalisis untuk mengetahui kualitas performa model antara lain yaitu: *Precision*, *Recall*, *F1-Score* dan *Mean Average Precision (mAP)*. Peneliti

menggunakan *tool* analisis *Confusion Matrix* dengan menghitung nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) [2].

Parameter-parameter tersebut dapat dihitung dengan beberapa persamaan. *Precision* merupakan perhitungan yang menunjukkan nilai ketetapan prediksi data positif yang benar [18], yang dihitung dengan persamaan 1.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall merupakan perhitungan yang mengukur seberapa baik model dapat mengidentifikasi semua sampel [18], yang dihitung dengan persamaan 2.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

F1-Score merupakan perhitungan yang menunjukkan nilai rata-rata harmonis dari *precision* dan *recall* [18], yang dihitung dengan persamaan 3.

$$F1 - Score = 2\frac{Precision\ .Recall}{Precision + Recall} \tag{3}$$

Terkahir, *Mean Average Precision (mAP)* merupakn perhitungan yang menunjukkan nilai performa keseluruhan model pada semua kelas yang diuji, yang membandingkan *bounding box* label dan *bounding box* deteksi [18]. Perhitungan ini didapat dengan menghitung rata-rata dari seluruh nilai *Average Precision (AP)* untuk setiap kelas, yang dihitung dengan persamaan 4.

$$mAP = \left(\frac{1}{N}\right) * \sum_{i=1}^{N} (AP_i) \tag{4}$$

Pada persamaan 4, simbol $\sum$ menyatakan penjumlahan dari nilai *AP* untuk setiap *class* yang diuji [2][14][6].Dengan memperhatikan hasil perhitungan *confusion matrix*, peneliti dapat mengevaluasi apakah model yang dihasilkan sudah memiliki performa yang baik dan layak digunakan untuk tahap implementasi, atau masih memerlukan proses optimasi lanjutan.

2.5. *Model Integration for OpenCV*

Model selanjutnya diintegrasikan ke dalam program berbasis OpenCV. OpenCV merupakan sebuah pustaka *Application Peripheral Interface (API)* yang dapat digunakan untuk pengolahan gambar serta bertujuan untuk menghadirkan *tools* dasar untuk *computer vision* [24]. Pada penelitian ini, OpenCV digunakan untuk mengakses kamera untuk membuka visi komputer, menampilkan video secara *real-time*, mengirimkan setiap *frame* video ke model untuk diproses, serta menampilkan hasil deteksi dari model dalam bentuk *bounding box*, label *class*, dan *confidence score* secara langsung pada tampilan video.

2.6. *Result Analysis*

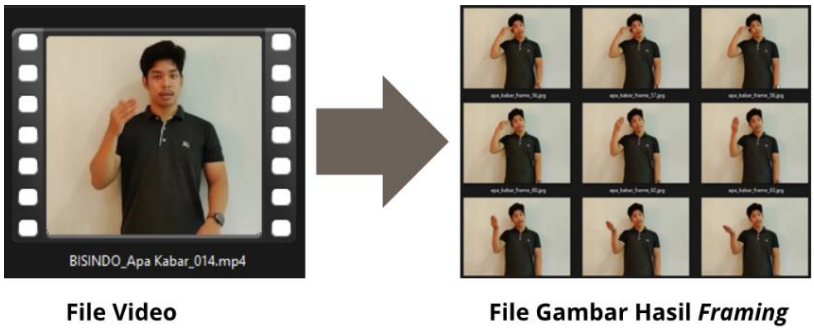
Tahap ini merupakan tahapan terakhir pada penelitin ini. Hasil dari model yang sudah diintegrasikan dengan OpenCV, selanjutnya dianalisis guna menarik kesimpulan. Penulis menggunakan beberapa parameter untuk menganalisis kecepatan, kesetabilan, dan kemampuan model dalam mengenalai *class* secara konsisten pada video *real-time*, yaitu: *Frame per Second (FPS)*, *Latency*, dan *Confidence Score*. *FPS* mengukur kecepatan model dalam memroses, *Latency* mengukur interaktivitas *delay* antara frame input dan output, sedangkan *Confidence Score* sebagai indikator pendukung dalam menunjukkan kemampuan serta konsistensi model dalam mengenali *class*.

Penulis tidak menggunakan *Confusion Matrix* sebagai parameter analisis performa model dalam mendeteksi secara *real-time*. Hal ini dikarenakan data *ground truth* tidak tersedia secara dinamis dalam pendeteksian secara *real-time*, sehingga nilai *Confusion Matrix* tidak dapat dihitung atau cenderung ambigu. Sehingga penggunaan *Confusion Matrix* pada implementasi secara *real-time* akan menghasilkan hasil analisis yang tidak valid.

3. HASIL DAN PEMBAHASAN

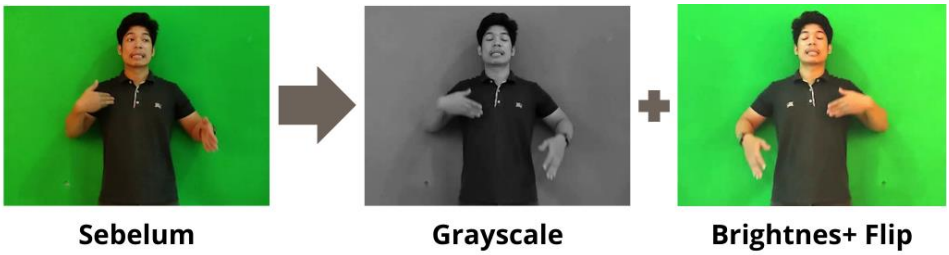
Hasil dan pembahasan dari setiap tahap penelitian yang dilakukan, disajikan pada bagian ini. Sebelum melatih model, semua data harus melalui tahap *preprocessing* seperti yang sudah dijelaskan pada metode

penelitian. Dilakukan *framing* untuk semua data video agar data dapat digunakan untuk pelatihan model. Contoh hasil *framing* dapat dilihat pada Gambar 2.



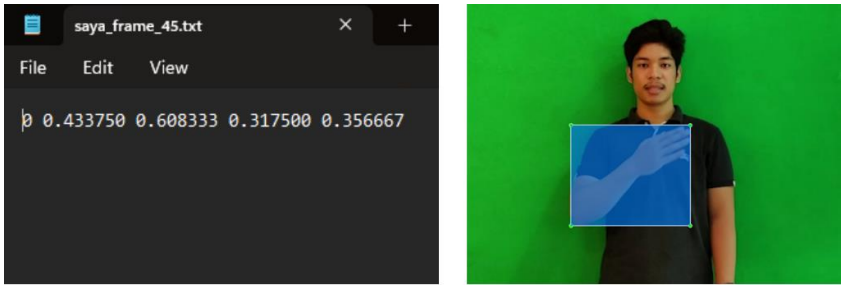
Gambar 2. Contoh hasil *framing*

Sebagian data gambar selanjutnya melalui tahap *augmentation*, dimana beberapa gambar diperbanyak jumlahnya atau digandakan dengan pencahayaan atau warna yang berbeda serta membalikkan gambar secara horizontal. Ini bertujuan untuk menambah variasi data gambar agar model tetap dapat mendeteksi objek dalam berbagai kondisi penchayaan dan kualitas kamera [11][14]. Contoh hasil *augmentation* dapat dilihat pada Gambar 3.



Gambar 3. Contoh hasil *augmentation*

Tahap terkahir dari *preprocessing* adalah *labeling*. Hasil dari tahap ini adalah sebuah *file* dengan format *.txt* berisi label untuk setiap data gambar. Dalam *file* label untuk setiap gambar, di dalamnya terdapat data yang menginformasikan indeks *class name*, koordinat tengah sumbu *x*, koordinat tengah sumbu *y*, panjang *bounding box* label, dan lebar *bounding box* label yang menunjukkan posisi dari setiap objek pada sebuah data gambar [2]. Contoh isi *file* label dan bentuk label pada *file* gambar dapat dilihat pada Gambar 4.

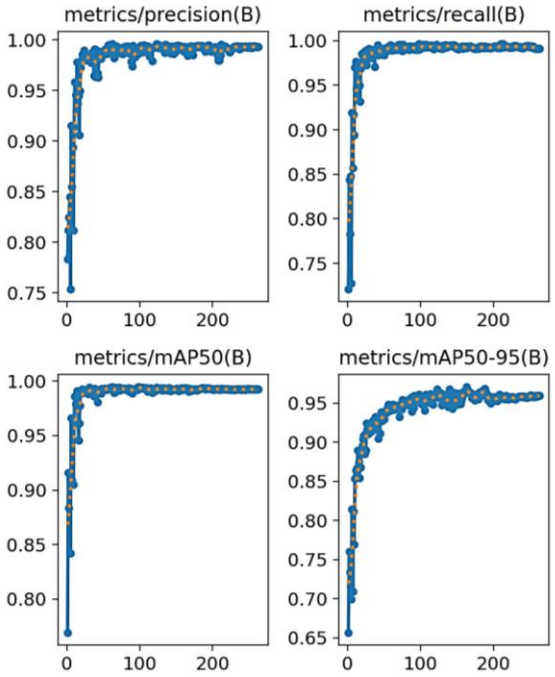


Gambar 4. Contoh data dalam file label (kiri), contoh posisi lebel pada data gambar (kanan)

Semua data *train* yang sudah melalui tahap *Preprocessing*, selanjutnya dilatih dengan teknik *transfer learning*. Pelatihan dilakukan dengan 263 *epoch*, yang berarti pelatihan dilakukan sebanyak 263 kali pengulangan penuh terhadap seluruh data pada *dataset*. Jumlah *epoch* ditentukan dengan teknik *early stopping*, yang berarti pelatihan akan berhenti otomatis ketika kondisi performa dalam pelatihan tidak menunjukkan kenaikan lagi dalam 100 *epoch* terakhir. Grafik metrik hasil pelatihan dapat dilihat pada Gambar 5.

Dari Gambar 5, hasil pelatihan model menunjukkan peningkatan pada beberapa parameter evaluasi. *Precision* dan *recall* meningkat dengan cepat hingga hampir 1,0 dalam 20 *epoch* pertama, hal ini menunjukkan rendahnya FP dan FN selama pelatihan. Nilai *mAP50* juga mengalami peningkatan dengan cepat mencapai hampir 1,0, menunjukkan akurasi deteksi yang tinggi pada *Intersection over Union (IoU)* ≥ 0,5. Sementara itu, *mAP50-95* meningkat lebih lambat tetapi tetap stabil hingga mendekati 0,95. Hal ini

menunjukkan bawha model dapat menentukan lokasi objek pada tingkat toleransi *IoU* yang berbeda dengan presisi yang tetap tinggi.



Gambar 5. Grafik metrik hasil pelatihan

Pengujian model dilakukan dengan data *test* yang telah disiapkan dalam *dataset* yang dikumpulkan dari empat partisipan. Hasil pengujian model menunjukkan nilai yang sangat tinggi dan stabil pada semua metrik evaluasi. Nilai rata-rata keseluruhan *class (All)* untuk *precision* adalah 0,983; *recall* 0,982; dan *F1-score* 0,982; yang menunjukkan bahwa model dapat mendeteksi objek dengan baik dan memiliki tingkat kesalahan yang rendah secara keseluruhan. Nilai *mAP50* sebesar 0,993 dan *mAP50-95* sebesar 0,938 menunjukkan bahwa *bounding box* prediksi yang dihasilkan memiliki tingkat kedekatan yang tinggi terhadap *ground truth*, baik pada ambang *IoU* yang rendah *(mAP50)* maupun tinggi *(mAP50-95)*.

Evaluasi per *class* juga menunjukkan performa model yang baik, walaupun terdapat sedikit kesalahan minor pendeteksian. *Class* "Marah" memiliki *precision* dan *recall* hampir sempurna (0,999 dan 1,0), serta *F1-score* 0,999. *Class* "Saya" memiliki *recall* sempurna (1,0), akan tetapi *precision* lebih rendah (0,965), yang menandakan adanya prediksi *False Positive*. Sebaliknya, *class* “Kamu” dan “Bingung” memiliki *precision* sempurna (1,0), tetapi *recall* sedikit lebih rendah yang mendandakan adanya prediksi *False Negative*. *Class* “Senang” memiliki *precision* dan *recall* yang tinggi (0,980 dan 0,992). *Class* "Apa Kabar" memiliki nilai *F1-score* paling rendah dibanding *class* lainnya (0,956), dengan *precision* dan *recall* yang seimbang (0,955 dan 0,958). Hal ini menunjukkan bahwa model lebih sering membuat kesalahan minor dalam memrediksi *class* tersebut dibandingkan dengan kelas lainnya, meskipun nilainya masih tergolong tinggi. Kesalahan minor ini kemungkinan disebabkan oleh bentuk gestur yang terlalu kompleks, atau terdapat kemiripan antar gestur[25].

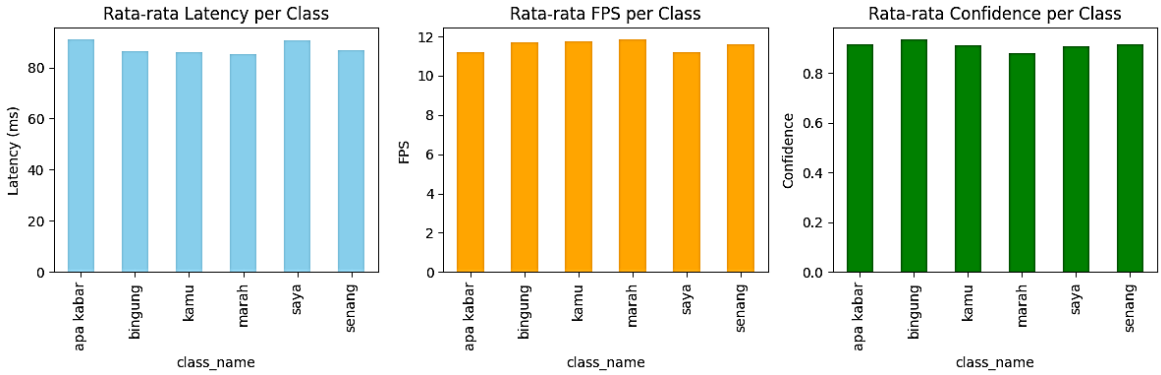
Secara keseluruhan, hasil ini menunjukkan bahwa model mampu mengenali gestur pada data pengujian dengan akurasi *bounding box* dan *precision* yang tinggi berdasarkan analisis perhitungan *Confusion Matrix*. Hasil pengujian dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian Model YOLO11

Class	Precision	Recall	F1-Score	mAP50	mAP50-95
All	0,983	0,982	0,982	0,993	0,938
Saya	0,965	1,000	0,982	0,995	0,863
Kamu	1,000	0,961	0,980	0,995	0,913
Senang	0,980	0,992	0,985	0,993	0,914
Bingung	1,000	0,979	0,989	0,995	0,968
Marah	0,999	1,000	0,999	0,995	0,949
Apa Kabar	0,955	0,958	0,956	0,985	0,921

Model selanjutnya diintegrasikan dengan OpenCV untuk implementasi pendeteksian secara *real-time*. Selama pengujian secara *real-time*, program akan melakukan pencatatan data yang digunakan untuk

menghitung parameter analisis seperti *FPS*, *Latency*, *Confidence Score*, dll ke dalam sebuah file CSV *(Comma Separated Values)*. Dengan *threshold* 0,75, data yang disimpan selanjutnya diolah untuk mengukur kecepatan, kesetabilan, dan kemampuan model dalam mengenali *class* secara konsisten. Terdapat 200 data per *class* yang digunakan untuk pengolahan data selanjutnya. Data yang diolah menghasilkan beberapa diagram seperti rata-rata *Latency*, *FPS*, dan *Confidence Score* per *class*, dapat dilihat pada Gambar 6.



Gambar 6. Diagram rata-rata *Latency*, *FPS*, dan *Confidence Score* per *class*

1. *Rata-rata Latency per Class*
   Pada Gambar 6, diagram *Rata-rata Latency per Class* (kiri) memvisualisasikan rata-rata waktu pemrosesan *(Latency)* dalam satuan milidetik *(ms)* untuk setiap *class* yang dideteksi. Diketahui bahwa diagram menunjukkan *latency* tertinggi untuk *class* “saya” dan “apa kabar”dengan nilai rata-rata ±90ms. Hal ini menjelaskan bahwa model membutuhkan waktu lebih lama untuk mendeteksi *class* tersebut dibandingkan dengan *class* lain seperti “marah” dan “kamu” dengan nilai rata-rata ±80ms, namun masih berada dibawah rata-rata normal (*latency* < 100) dan stabil. Hal ini dapat terjadi karena gerakan tangan pada *class* tersebut lebih kompleks dan/atau sulit untuk dikenali oleh model.

2. *Rata-rata FPS per Class*
   Pada Gambar 6, diagram *Rata-rata FPS per Class* (tengah) menampilakn rata-rata *FPS* saat model mendeteksi setiap *class*. Dapat dilihat bahwa *FPS* relatif stabil dan tinggi dengan rata-rata ±11,5 di semua *class*. Hal ini menandakan bahwa pendeteksian berjalan dengan lancar di setiap *class* yang terdeteksi.

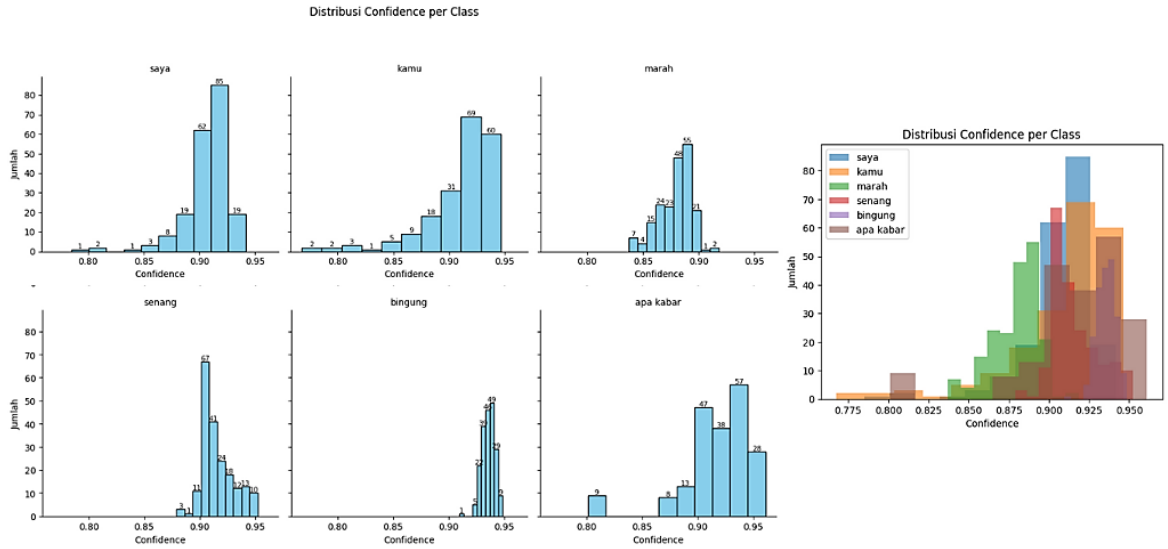
3. *Rata-rata Confidence per Class*
   Pada Gambar 6, diagram *Rata-rata Confidence per Class* (kanan) menunjukkan nilai rata-rata *confidence score* (tingkat keyakinan) untuk setiap *class*. Diagram menunjukkan bahwa rata-rata *confidence score* tinggi dan konsisten di seluruh *class* (±0,9 atau 90%). Hal ini menunjukkan bahwa model cukup yakin dengan prediksinya. Gambar 7 berikut memperlihatkan distribusi *confidence score* untuk setiap *class*.

Dua diagram pada Gambar 7 merupakan diagram Distribusi *Confidence per Class* yang memperlihatkan sebaran *confidence score* untuk setiap *class*. Diagram sebelah kiri memperlihatkan gambaran yang lebih detail dengan menggambarkan distribusi pada masing-masing *class* secara terpisah. Diagarm ini menjelaskan bahwa pada *class* "saya", "kamu", "senang", “apa kabar”, dan "bingung”, memiliki sebaran yang padat dan terkonsentrasi di atas 0,9. Artinya model cukup konsisten dalam mendeteksi *class* tersebut. Sedangkan *class* “marah” memiliki distribusi yang lebih lebar dan cenderung lebih rendah dengan *confidence score* < 0,9, namun masih tergolong tinggi karena *confidence score* stabil di atas 0,85. Hal ini dapat terjadi karena beberapa hal, salah satunya adalah posisi tangan yang sedikit berbeda dari data pelatihan.

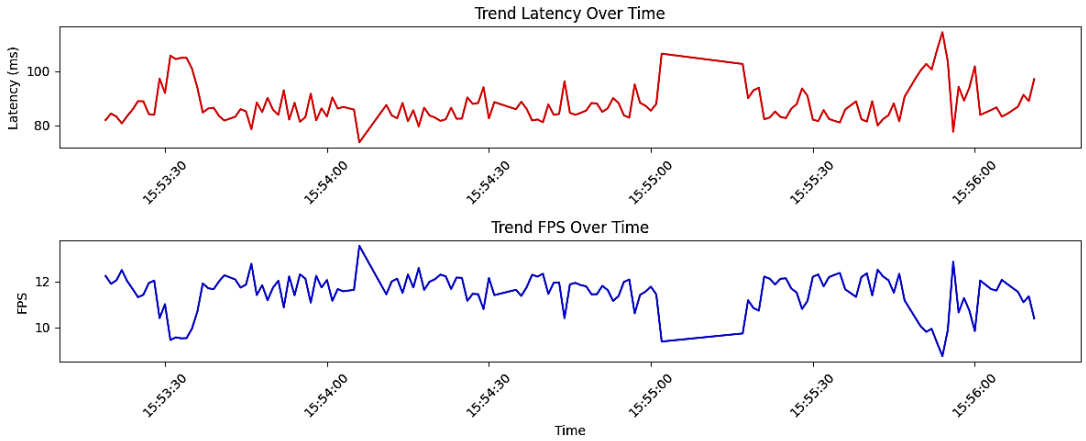
File CSV yang berisi 200 data per *class* yang dicatat sebelumnya, juga diolah lebih lanjut untuk menganalisis *Trend Latency* dan *Trend FPS*. *Trend Latency* digunakan untuk mengetahui kecepatan dan stabilitas model dalam mendeteksi objek. Ini penting guna memastikan model bekerja dengan efisien dan responsif. *Trend FPS* berguna untuk menunjukkan tingkat kelancaran model dalam mendeteksi objek secara *real-time*, yang merupakan indikator utama untuk menilai efisiensi dan responsifitas model. Hasil pengolahan ini ditunjukkan pada Gambar 8.

Pada Gambar 8, grafik *Trend Latency Over Time* (garis merah) menunjukkan *latency* selama proses pendeteksian secara *real-time*. Terdapat beberapa peningkatan *latency* selama pendeteksian, namun rata-rata stabil di rentang 80-100ms. Peningkatan *latency* yang terjadi dapat disebabkan oleh beberapa hal, antara lain

kompleksitas gerakan tangan pada *class* tertentu, terdapat gerakan tangan yang sulit dideteksi, atau penggunaan CPU yang terlalu tinggi.



Gambar 7. Diagram distribusi *condifence score* per *class*



Gambar 8. Grafik *Trend Latency* dan *Trend FPS*

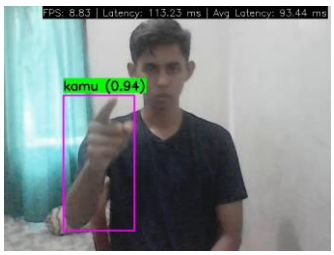
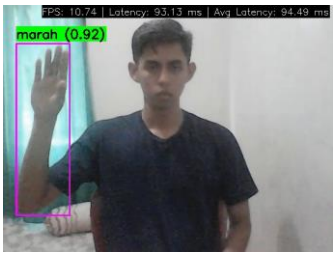
Pada Gambar 8, grafik *Trend FPS Over Time* (garis biru) menunjukkan *frame rate* selama proses pendeteksian secara *real-time*. Dapat dilihat bahwa *FPS* relatif stabil di kisaran 11-12FPS, yang menunjukkan performa model cukup baik dalam mendeteksi objek. Terdapat beberapa penurunan *FPS* yang kemungkinan besar disebabkan oleh peningkatan *latency*, karena grafik *FPS* berbanding terbalik dengan grafik *latency* dimana semakin tinggi *latency* maka semakin rendah *FPS*.

Selama pengujian *real-time*, dilakukan validasi untuk mengevaluasi keterbatasan model dalam mendeteksi gestur sesuai dengan *class* frasa yang dilatihakan. Validasi dilakukan dengan menguji keenam frasa gestur pada integrasi *real-time*. Hasil menunjukkan bahwa model mampu mengenali dan membedakan masing-masing gestur frasa dengan baik, meskipun masih terdapat beberapa kesalahan minor. Namun demikian, ditemukan beberapa kasus di mana model secara keliru mendeteksi gestur acak yang tidak dilatihkan, sebagai salah satu *class* frasa yang paling mirip dalam *dataset*. Beberapa hasil pendeteksian termasuk kesalahan deteksi yang ditemukan disajikan pada Tabel 3.

Berdasarkan Tabel 3, ditemukan tiga kasus kesalahan selama evaluasi secara *real-time*. Dua di antaranya adalah gestur yang tidak dilatihkan (“L” dan “Metal”), namun terdeteksi sebagai salah satu gestur frasa yang dilatihkan, yaitu “kamu” dan “marah”. Satu kasus lainnya merupakan gestur frasa yang sebenarnya dilatihkan, yaitu “senang”, namun sering terdeteksi sebagai gestur frasa lainnya, yaitu “apa kabar”. Hasil visual dari kesalahan tersebut dapat dilihat pada Gambar 15, 16, dan 17 dalam Tabel 3. Kesalahan ini kemungkinan disebabkan karena kemiripan antara gestur *input* dan *output*. Selain itu, kekurangan juga muncul pada beberapa skenario, misalnya ketika warna dominan pakaian pengguna serupa dengan warna latar belakang sehingga model kesulitan memprediksi *bounding box*. Kekurangan lain terjadi

jika pengguna mengenakan pakaian berlengan panjang, karena sebagian besar lengan tertutup, yang membuat model sulit mendeteksi beberapa kelas gestur seperti “saya”, “apa kabar”, dan “senang”. Kekurangan ini menyebabkan prediksi letak *bounding box* tidak akurat atau cenderung melebar, serta kesalahan dalam memprediksi *class* frasa, seperti yang ditunjukkan pada Gambar 18.

Tabel 3. Hasil Evaluasi Integrasi *Real-Time*

No	Gestur Masukan	Jenis Gestur	Keluaran Visual	Keluaran Teks	Status
1	Saya	Dilatihkan		Saya	TP
2	Kamu	Dilatihkan		Kamu	TP
3	Senang	Dilatihkan		Senang	TP
4	Bingung	Dilatihkan		Bingung	TP
5	Marah	Dilatihkan		Marah	TP

Gambar 9. Hasil evaluasi gestur saya

Gambar 10. Hasil evaluasi gestur kamu

Gambar 11. Hasil evaluasi 1 gestur senang

Gambar 12. Hasil evaluasi gestur bingung

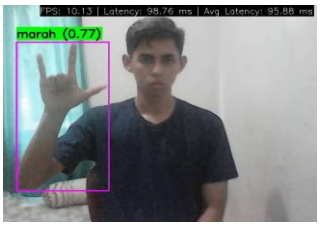
Gambar 13. Hasil evaluasi gestur marah

No	Gestur Masukan	Jenis Gestur	Keluaran Visual	Keluaran Teks	Status
6	Apa Kabar	Dilatihkan		Apa Kabar	TP

Gambar 14. Haisl evaluasi gestur apa kabar

7	Gestur “L”	Tidak Dilatihkan		Kamu	FP

Gambar 15. Hasil evaluasi gestur L

8	Gestur Metal	Tidak Dilatihkan		Marah	FP

Gambar 16. Hasil evaluasi gestur metal

9	Senang	Dilatihkan		Apa Kabar	FP

Gambar 17. Hasil evaluasi 2 gestur senang

Dari Gambar 18 di atas dapat dilihat beberapa kesalahan ketika mengenakan pakaian lengan panjang dengan warna serupa dengan *background*. Ketika mempraktikkan gestur “saya” (kiri) model tidak mengenali gestur. Ketika mempraktikan gestur “senang” (tengah) model sulit dalam menentukan *bounding box*. Ketika mempraktikkan gestur “apa kabar” (kanan) hasil *bounding box* prediksi model sedikit melebar dan *confidence score* menurun. Kekurangan ini terjadi karena tidak adanya data gambar pada *dataset* yang menggnakan pakaian lengan panjang dan warna serupa dengan background, sehingga model belum terlalu mempelajari dan mengenali kesalahan pada skenario tersebut. Meskipun demikian, dapat diamati bahwa setiap kesalahan yang terjadi memiliki *confidence score* yang lebih rendah dibandingkan dengan hasil deteksi yang benar, seperti yang divisualkan pada Gambar 9 hingga 14 dalam Tabel 3. Kondisi ini menunjukkan bahwa model telah terlatih dengan cukup baik, tidak *overfiting* maupun *underfiting*, serta dapat mengenali pola karakteristik dari setiap *class* gestur. Secara umum, model menunjukkan performa yang baik dan berhasil mendeteksi enam frasa BISINDO yang dilatihkan secara *real-time* dengan konsisten.

Gambar 18. Hasil ketika menggunakan pakaian panjang dan warna sama dengan *background*

4. DISKUSI

Hasil menunjukkan bahwa model YOLO11, yang diintegrasikan dengan OpenCV, mampu mengenali enam frasa BISINDO secara *real-time* dengan akurasi 0,983; recall 0,982; dan mAP50 0,993. Nilai-nilai ini menunjukkan tingkat keandalan dan konsistensi yang tinggi dalam mendeteksi berbagai kelas frasa, termasuk stabilitas dalam pengujian *real-time* dengan *latency* rata-rata <100ms dan *frame rate* sekitar 11–12FPS. Kecepatan dan akurasi ini mendukung pemilihan YOLO11 sebagai solusi *real-time* yang memenuhi kebutuhan sistem komunikasi bahasa isyarat secara langsung dengan latency yang relatif rendah. Temuan ini sejalan dengan kinerja YOLO11 pada penelitian yang dilakukan oleh Alsharif et al. (2025) dalam pengenalan gestur ASL, seperti yang dilaporkan dalam penelitian tentang pengenalan ASL *real-time* (*mAP50* ≈ 98,2%) [26]. Hal ini menunjukkan bahwa YOLO11 dapat mempertahankan akurasi tinggi dengan responsifitas tetap. Implementasi ini memperkuat penggunaan YOLO11 sebagai dasar untuk sistem pengenalan bahasa isyarat yang dapat langsung digunakan dalam aplikasi pendukung komunikasi di masyarakat, seperti pendidikan inklusif dan layanan public lainnya.

5. KESIMPULAN

Penelitian ini berhasil membangun program untuk pengenalan frasa bahasa isyarat tangan BISINDO secara *real-time* dengan mengintegrasikan algoritma YOLO11 dan *library* OpenCV. Model pada program ini dilatih menggunakan lebih dari 3.000 data gambar yang mewakili enam *class* frasa yang berbeda. Hasil pengujian menunjukkan bahwa model memiliki rata-rata nilai *precision* dan *recall* di atas 0,98; *F1-Score* sebesar 0,982; *mAP50* sebesar 0,993; dan *mAP50-95* sebesar 0,938. Secara *real-time*, sistem dapat beroperasi secara stabil dengan *latency* rata-rata 80-90ms dan *frame rate* 11-12 FPS, serta *confidence score* rata-rata 0,9 untuk semua *class*. Namun, Sistem ini masih memiliki keterbatasan dalam beberapa skenario, terutama ketika pengguna mengenakan pakaian berlengan panjang atau pakaian yang warnanya mirip dengan latar belakang, yang mengakibatkan penurunan akurasi *bounding box* dan peningkatan kesalahan prediksi *class*, serta model sulit membedakan gestur *input* dan *output* yang secara bentuk serupa. Hasil ini membuktikan bahwa integrasi YOLO11 dan OpenCV berhasil digunakan sebagai algoritma dalam pengenalan frasa bahasa isyarat tangan BISINDO secara *real-time*. Hal ini dibuktikan dengan eksperiman pengenalan enam frasa berbeda, yaitu: "saya", "kamu", "senang", "bingung", "marah", dan "apa kabar". Hasil ini juga dapat dikembangkan lagi untuk pengenalan frasa yang lebih lengkap, serta diuji oleh pengguna sesuangguhnya guna mengukur kegunaan sistem dalam scenario dunia nyata. Selain itu, hasil ini juga dapat dikembangkan untuk diimplementasikan pada beberapa platform seperti *mobile device* agar menciptakan fungsionalitas yang lebih terintegrasi atau *web browser* agar lebih fleksibel, untuk mendukung komunikasi yang lebih interaktif dan responsif bagi penyandang tunarungu dan tunawicara.

REFERENSI

[1] E. Libra Kelana, M. Riko, A. Prasetya, dan M. Zulfadhilah, "Integrating the CNN Model with the Web for Indonesian Sign Language (BISINDO) Recognition," Jun 2025. [Daring]. Tersedia pada: http://jurnal.polibatam.ac.id/index.php/JAIC

[2] A. Sani dan S. Rahmadinni, "Deteksi Gestur Tangan Berbasis Pengolahan Citra," *Jurnal Rekayasa Elektrika*, vol. 18, no. 2, hlm. 115–124, Jul 2022, doi: 10.17529/jre.v18i2.25147.

[3] E. Altiarika dan W. P. Sari, "Pengembangan Deteksi Realtime untuk Bahasa Isyarat Indonesia dengan Menggunakan Metode Deep Learning Long Short Term Memory dan Convolutional Neural Network," *Jurnal Teknologi Informatika dan Komputer*, vol. 9, no. 1, hlm. 1–13, Mar 2023, doi: 10.37012/jtik.v9i1.1272.

[4] A. Al-Shaheen, M. Çevik, dan A. Alqaraghuli, "American Sign Language Recognition using YOLOv4 Method," *International Journal of Multidisciplinary Studies and Innovative Technologies*, vol. 6, no. 1, hlm. 61–65, Jun 2022, doi: 10.36287/ijmsit.6.1.61.

[5] H. Deshpande, A. Singh, dan H. Herunde, "Comparative Analysis on YOLO Object Detection with OpenCV," *International Journal of Research in Industrial Engineering*, vol. 9, hlm. 46–64, Mar 2020, doi: 10.22105/riej.2020.226863.1130.

[6] A. Sharma, V. Kumar, dan L. Longchamps, "Comparative performance of YOLOv8, YOLOv9, YOLOv10, YOLOv11 and Faster R-CNN models for detection of multiple weed species," *Smart Agricultural Technology*, vol. 9, hlm. 1–14, Des 2024, doi: 10.1016/j.atech.2024.100648.

[7] R. Khanam dan M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," hlm. 1–9, Okt 2024, [Daring]. Tersedia pada: http://arxiv.org/abs/2410.17725

[8] T. Cut Al-Saidina Zulkhaidi, E. Maria, P. Studi Teknologi Rekayasa Perangkat Lunak, dan P. Pertanian Negeri Samarinda, "Pengenalan Pola Bentuk Wajah dengan OpenCV," *JURTI*, vol. 3, no. 2, hlm. 181–186, 2019.

[9] I. Inayatul Arifah, F. Nur Fajri, dan G. Qorik Oktagalu Pratamasunu, "Deteksi Tangan Otomatis Pada Video Percakapan Bahasa Isyarat Indonesia Menggunakan Metode YOLO Dan CNN," *Journal of*

Applied Informatics and Computing (JAIC), vol. 6, no. 2, hlm. 171–176, Nov 2022, [Daring]. Tersedia pada: http://jurnal.polibatam.ac.id/index.php/JAIC

[10] S. Daniels, "Pengenalan Bahasa Isyarat pada Data Video Menggunakan Metode CNN dengan Arsitektur YOLO," Surabaya, Jul 2020.

[11] A. Mujahid *dkk.*, "Real-time hand gesture recognition based on deep learning YOLOv3 model," *Applied Sciences (Switzerland)*, vol. 11, no. 9, hlm. 1–15, Mei 2021, doi: 10.3390/app11094164.

[12] A. Hasyim Nur'azizan, A. Riqza Ardiansyah, dan R. Fernandis, "Implementasi Deteksi Bahasa Isyarat Tangan Menggunakan OpenCV dan MediaPipe," dalam *Prosiding Seminar Nasional Teknologi Dan Sains Tahun 2024, Vol. 3*, Kediri, Jan 2024, hlm. 331–337.

[13] N. Renaningtias, F. Putra Utama, A. Nur, dan A. Sobri, "Deteksi Bahasa Isyarat Indonesia (BISINDO) Pada Video dengan YOLOv7," *JSAI: Journal Scientific and Applied Informatics*, vol. 8, no. 1, hlm. 1–8, Jan 2025, doi: 10.36085.

[14] A. I. Pradana, H. Harsanto, dan W. Wijiyanto, "Deteksi Rambu Lalu Lintas Real-Time di Indonesia dengan Penerapan YOLOv11: Solusi Untuk Keamanan Berkendara," *Jurnal Algoritma*, vol. 21, no. 2, hlm. 145–155, Nov 2024, doi: 10.33364/algoritma/v.21-2.2106.

[15] P. Gidion Bagas, A. Hagi Azzam, dan R. Chaerur, "Deteksi Dan Pengenalan Gestur Tangan Secara Real-Timemenggunakan Jaringansaraf Tiruan Konvolusional," *METHODIKA*, vol. 9, hlm. 30–34, Sep 2023.

[16] A. Gupta Aditya Verma dan A. M. Yadav Arvidhan BCA-IOP Associate Professor, "YOLO OBJECT DETECTION USING OPENCV," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 10, hlm. 233–237, Feb 2021, [Daring]. Tersedia pada: http://www.ijeast.com

[17] A. B. Aziz *dkk.*, "YOLO-V4 Based Detection of Varied Hand Gestures in Heterogeneous Settings," dalam *Proceedings of the 2024 International Conference on Artificial Intelligence (AII 2023)*, Springer Science and Business Media Deutschland GmbH, Agu 2024, hlm. 1–14. doi: 10.1007/978-3-031-68639-9_21.

[18] V. S. Henry, Y. P. Sumihar, dan F. Maedjaja, "Automated Anaemia Detection From Conjunctiva Images : A Machine Learning Approach For Android Application," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 9, no. 2, hlm. 798–805, Mei 2024, doi: 10.29100/jipi.v9i2.5218.

[19] M. Juan, S. Alwin, dan S. M. L. Arie, "Hand Gesture Detection Application in Sign Language," *Jurnal Teknik Informatika*, vol. 17, hlm. 285–296, Sep 2022, Diakses: 2 Mei 2025. [Daring]. Tersedia pada: https://ejournal.unsrat.ac.id/index.php/informatika

[20] J. Huang, K. Wang, Y. Hou, dan J. Wang, "LW-YOLO11: A Lightweight Arbitrary-Oriented Ship Detection Method Based on Improved YOLO11," *Sensors*, vol. 25, no. 1, hlm. 1–18, Jan 2025, doi: 10.3390/s25010065.

[21] W. Kurniawan, A. Kurniasih, dan M. A. Ghani, "Real or Deepfake Face Detection in Images and Video Data using YOLO11 Algorithm," *Journal of Artificial Intelligence and Engineering Applications*, vol. 4, no. 2, hlm. 2808–4519, Feb 2025, [Daring]. Tersedia pada: https://ioinformatic.org/

[22] S. Dede Haris, I. Bahtiar, dan Juhartini, "Object Detection Untuk Mendeteksi Citra Buah Buahan Menggunakan Metodeyolo," *Jurnal Kecerdasan Buatan dan Teknologi Informasi*, vol. 2, no. 2, hlm. 70–80, Mei 2023.

[23] M. Alaftekin, I. Pacal, dan K. Cicek, "Real-time sign language recognition based on YOLO algorithm," *Neural Comput Appl*, vol. 36, no. 14, hlm. 7609–7624, Mei 2024, doi: 10.1007/s00521-024-09503-6.

[24] D. Khairianto dan R. Firdaus, "Penerapan Hand Gesture Recognition Sebagai Media Kontrol Presentasi Aplikasi Powerpoint," *Jurnal Mahasiswa Teknik Informatika*, vol. 8, no. 2, hlm. 1852–1860, Apr 2024.

[25] I. Gusti, A. Made, Y. Mahaputra, P. Alit, W. Santiary, dan I. Ketut Swardika, "Rancang Bangun Penerjemah BISINDO Real-time Berbasis Kamera dan Deep Learning dengan Kendali Suara ESP32 WiFi," *Jurnal Elektro dan Mesin Terapan*, vol. 11, no. 1, hlm. 33–42, 2025, doi: 10.35143/elementer.v11i1.

[26] B. Alsharif, E. Alalwany, A. Ibrahim, I. Mahgoub, dan M. Ilyas, "Real-Time American Sign Language Interpretation Using Deep Learning and Keypoint Tracking," *Sensors*, vol. 25, no. 7, hlm. 1–21, Mar 2025, doi: 10.3390/s25072138.