INSECT: Jurnal Teknik Informatika
INFORMATICS AND SECURITY e-ISSN. 2614-431X

Studi Perbandingan Algoritma YOLO dan FOMO untuk *Object Detection* pada Perangkat ESP32-CAM

Firdaus*[1], Mahardika Wibowo[2], Rahmat Tullah[3], Wieke Ricesa[4]
[1,2,3,4]Prodi Teknik Informatika, Institut Teknologi dan Bisnis Bina Sarana Global
E-mail: [1*]**firdaustegal93@gmail.com**, [2]mahardikawibowo11@gmail.com,
[3]rahmattullah@global.ac.id, [4]wricesa@gmail.com

Abstrak

Penelitian ini membandingkan performa dua algoritma pendeteksian objek, YOLO (You Only Look Once) dan FOMO (Faster Objects, More Objects), pada ESP32-CAM dengan sumber daya yang lebih rendah. Meskipun YOLO sering digunakan dalam aplikasi pendeteksian objek, namun penggunaannya pada perangkat dengan performa komputer yang rendah, seperti ESP32-CAM, masih terbatas. Sementara FOMO, yang dirancang untuk perangkat dengan keterbatasan komputasi, dianggap mampu memberikan alternatif yang lebih efektif. Penelitian ini mengevaluasi kinerja kedua algoritma berdasarkan tiga parameter utama: akurasi deteksi, waktu inferensi, dan penggunaan memori RAM. Penemuan penelitian menunjukkan, FOMO memiliki keunggulan signifikan dibandingkan YOLOv5 Nano dalam hal waktu inferensi, yang lebih cepat hingga beberapa kali lipat. Meskipun FOMO menghasilkan F1 Score yang sangat tinggi (99,2%), perbandingan akurasi menggunakan metric mAP tidak dapat dilakukan dengan FOMO. YOLOv5 Nano, di sisi lain, memiliki akurasi yang lebih baik pada mAP tetapi membutuhkan waktu inferensi yang lebih lama dan penggunaan memori yang lebih besar. Tujuan dari penelitian ini adalah untuk memastikan bahwa FOMO kekurangan di area lain dan YOLOv5 Nano dapat dioptimalkan lebih lanjut, terutama dalam deteksi objek dan waktu inferensi yang lebih efisien di perangkat dengan sumber daya terbatas.

Kata kunci*—YOLOv5 Nano, FOMO, ESP32-CAM, Edge impulse, Object Detection*

1. PENDAHULUAN

Penelitian tentang penerapan algoritma pendeteksian objek pada perangkat dengan sumber daya terbatas, khususnya ESP32-CAM, masih terbatas. Meskipun algoritma seperti YOLO telah banyak diaplikasikan dalam pendeteksian objek [1], namun aplikasinya pada perangkat dengan keterbatasan komputasi, seperti ESP32-CAM, masih belum banyak dibahas. Modul ESP32-CAM yang merupakan kamera berbasis mikrokontroler ESP32 sering digunakan dalam pengembangan sistem Edge Computing [2] karena memiliki ukuran yang kecil, biaya yang murah, dan konsumsi daya yang tinggi. Namun, keterbatasan memori dan kekuatan pemrosesan perangkat ini menjadi tantangan utama dalam menjalankan algoritma pendeteksian objek yang kompleks.

Sebagian besar algoritma pendeteksian objek modern membutuhkan waktu komputasi yang besar. Oleh karena itu, pemilihan algoritma yang efisien penting agar sistem dapat beroperasi dengan baik pada perangkat seperti ESP32-CAM. Salah satu algoritma alternatif yang ringan adalah FOMO, dirancang untuk perangkat dengan kapasitas komputasi terbatas [3]. Tujuan dari penelitian ini adalah untuk mengetahui seberapa baik varian YOLOv5 Nano dan FOMO bekerja dalam mendeteksi objek pada ESP32-CAM. Penelitian ini berfokus pada tiga aspek utama: akurasi deteksi, kecepatan pemrosesan (waktu inferensi), dan konsumsi sumber daya.

ESP32-CAM digunakan sebagai objek penelitian perbandingan karena perangkat mikrokontroler ini mudah diperoleh dan telah banyak digunakan dalam penelitian yang berhubungan dengan pengambilan gambar. Selain itu, perangkat ini memiliki kompatibilitas dengan platform *Edge Impulse*, yang memungkinkan penelitian dapat dilakukan dengan akurat. Demikian, ESP32-CAM cocok dijadikan objek penelitian perbandingan algoritma FOMO dan YOLOv5 Nano.

Penelitian sebelumnya, seperti yang dilakukan oleh Firdaus Jamaludin dkk. (2024), mengimplementasikan YOLOv8 dalam deteksi objek, namun tidak menguji penerapannya pada perangkat dengan sumber daya terbatas seperti ESP32-CAM [4]. Penelitian ini mengadaptasi pendekatan tersebut dan membandingkan YOLO dengan FOMO dalam konteks perangkat berbasis *Edge Impulse* [5], dengan harapan dapat memberikan wawasan mengenai algoritma deteksi objek yang lebih efisien pada perangkat dengan keterbatasan sumber daya.

Dengan demikian, tujuan penelitian ini adalah untuk memberikan rekomendasi mengenai algoritma yang paling cocok digunakan pada perangkat dengan sumber daya terbatas, khususnya untuk pengembangan sistem berbasis visi komputer yang lebih efisien.

2. METODE PENELITIAN

2.1.Desain Penelitian

Penelitian ini bertujuan untuk melakukan evaluasi komparatif terhadap kinerja dua algoritma deteksi objek, yaitu YOLO *(You Only Look Once)* dan FOMO (*Faster Objects, More Objects),* pada perangkat ESP32-CAM yang memiliki keterbatasan sumber daya komputasi. Pendekatan eksperimental akan diterapkan dengan membandingkan parameter kinerja meliputi waktu inferensi, penggunaan RAM, dan akurasi deteksi objek dalam lingkungan pengujian terkontrol.

2.2. Alat dan Bahan

Penelitian ini menggunakan kombinasi perangkat keras dan perangkat lunak sebagai berikut:

Perangkat keras: ESP32-CAM

Perangkat lunak: *Arduino* IDE dan *Edge Impulse*

2.3. Desain Eksperimen

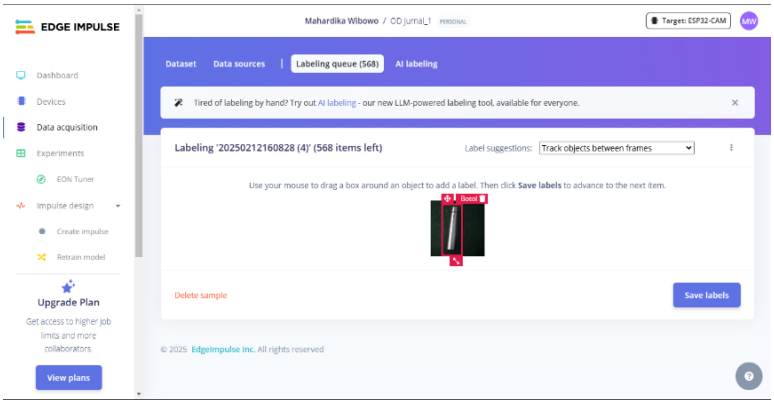
2.3.1 Pengumpulan Data

Kumpulan data yang digunakan dalam penelitian ini terdiri dari gambar objek yang telah dilabeli, termasuk botol, mainan, dan wajah. Data gambar dikumpulkan melalui kamera ESP32-CAM di berbagai kondisi pencahayaan dan sudut pandang untuk memastikan variasi yang cukup dalam pelatihan model.

2.3.2 Pelatihan Model

Proses pelabelan data di *platform Edge Impulse*, yang digunakan untuk melatih model kecerdasan buatan berbasis *computer vision*. Dalam proses ini, setiap objek dalam gambar, seperti botol, diberi label secara manual untuk membangun *dataset* yang akan digunakan dalam pelatihan model. Fitur AI *labeling* dengan LLM juga tersedia untuk membantu proses pelabelan secara otomatis dan meningkatkan efisiensi. *dataset* yang telah dilabeli kemudian digunakan untuk melatih model FOMO *(Faster Objects, More Objects)* dan YOLOv5 Nano. FOMO adalah model ringan yang dirancang untuk deteksi objek pada perangkat dengan daya komputasi rendah, sedangkan YOLOv5 Nano merupakan varian terkecil dari YOLO yang tetap mampu melakukan deteksi objek secara *realtime*. Proses pelatihan dilakukan langsung melalui *Edge Impulse,* dan hasilnya dianalisis untuk meningkatkan akurasi model. Target perangkat yang digunakan dalam penelitian ini adalah ESP32-CAM, yang menunjukkan bahwa model yang dikembangkan harus

ringan dan efisien agar dapat berjalan dengan baik pada perangkat dengan keterbatasan sumber daya. Setelah pelabelan selesai, *dataset* ini akan digunakan untuk mengoptimalkan kinerja model dalam mendeteksi objek secara lebih akurat dan cepat. Berikut adalah tampilan pelebelan dari *Edge Impulse* pada gambar 1



Gambar 1 Melakukan Pelabelan

2.3.3. Pengaturan Pelatihan FOMO

1 Jumlah Siklus Pelatihan

Jumlah siklus pelatihan diatur 60 siklus, untuk menentukan seberapa sering model akan diterapkan pada *dataset*. Semakin banyak siklus, model memiliki kapasitas yang lebih besar untuk belajar dari *dataset*, tetapi membutuhkan perhatian yang konstan. Banyak siklus dapat menyebabkan *overfitting* atau kondisi model terlalu menyesuaikan dengan data pelatihan, sehingga kinerjanya buruk pada data baru.

2 Tingkat Pembelajaran

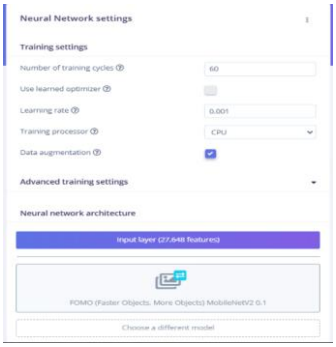
Ambang batas pembelajaran disesuaikan menjadi 0,001, dimana parameter ini menjadi kunci untuk mengendalikan seberapa besar langkah yang diambil dalam pembaruan bobot. Hal ini berkaitan dengan laju pembelajaran yang terlalu tinggi sehingga dapat membuat model kehilangan solusi optimal, sementara laju pembelajaran yang terlalu rendah dapat membantu pelatihan.

3 Augmentasi Data

Pengaktifan metode augmentasi data digunakan untuk meningkatkan variasi dalam *dataset* pelatihan. Dengan lebih banyak contoh yang berbeda dengan menambahkan *dataset* virtual menggunakan metode rotasi, *flipping*, dan perubahan kontras gambar, diharapkan dapat mengurangi kemungkinan *overfitting* dan membantu model algoritma deteksi objek belajar lebih baik.

4 Model yang Digunakan

FOMO *(Faster Objects, More Objects)* adalah model yang dipilih karena arsitekturnya dirancang untuk mendeteksi objek dengan efisiensi tinggi, sehingga membuatnya cocok untuk aplikasi yang dimana kecepatan dan akurasi sangat penting. Model pelatihan pada gambar 2



Gambar 2 Pengaturan Pelatihan FOMO

2.3.4. Pengaturan Pelatihan YOLOv5 Nano

1 Jumlah Siklus Pelatihan

Jumlah siklus pelatihan diatur ke 70 siklus. Untuk menunjukkan bahwa model dilatih selama 70 siklus berulang melalui *dataset*. Jumlah siklus ini penting untuk memastikan model dapat belajar dengan baik, tetapi juga harus diperhatikan agar tidak terjadi *overfitting.*

2 Tingkat Pembelajaran

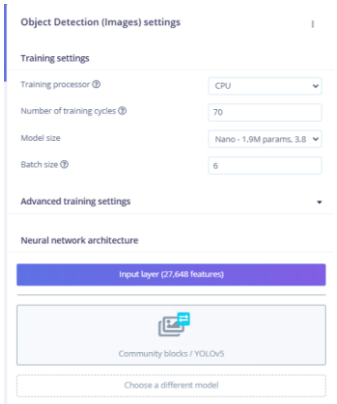
Meskipun tidak terlihat dalam gambar 2, tingkat pembelajaran adalah parameter yang sangat penting dalam pelatihan model. Ini mengontrol seberapa besar langkah yang diambil dalam pembaruan bobot selama pelatihan. Tingkat pembelajaran yang tepat membantu model mencapai konvergensi yang baik.

3 Augmentasi Data

Seperti yang ditunjukkan pada gambar 2, *augmentasi* data adalah cara untuk meningkatkan variasi dalam kumpulan data pelatihan. Meskipun tidak disebutkan secara eksplisit, *augmentasi* data dapat menggunakan metode rotasi, *flipping*, dan perubahan kontras gambar untuk membantu model belajar lebih baik dan meningkatkan *generalisasi.*

4 Model yang Digunakan

Model yang digunakan dalam pengaturan ini adalah blok Komunitas YOLOv5 Nano dengan ukuran Nano. Model ini memiliki 1,9 juta parameter dan ukuran yang relatif kecil, sehingga cocok untuk aplikasi yang membutuhkan kecepatan dan efisiensi tinggi. YOLOv5 Nano dikenal dengan kemampuannya untuk mendeteksi objek dengan cepat dan akurat. Model pelatihan pada gambar 3.



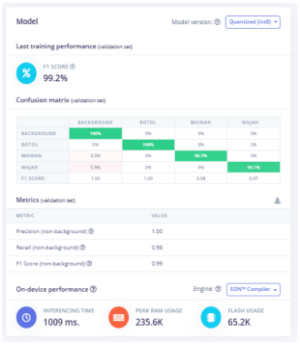
Gambar 3 Pengaturan Pelatihan YOLOv5 Nano

2.3.4.1 Perbedaan Pelatihan FOMO dan YOLOv5 Nano

1. Jumlah Siklus Pelatihan
   FOMO: 60 siklus
   YOLOv5 Nano: 70 siklus
   Analisis : Untuk jumlah siklus pelatihan sesuai dengan algoritma YOLOv5 Nano, dan FOMO. Nilai siklus pengulangan diambil karena dari hasil uji coba yang telah dilakukan, nilai tersebut mendapatkan F1 Score yang tinggi dibandingkan nilai pengulangan yang lain.
2. Pengolah Pelatihan
   FOMO: CPU
   YOLOv5 Nano: CPU
   Analisis: Keduanya menggunakan CPU sebagai prosesor untuk pelatihan. CPU lebih mudah diakses oleh pengguna yang memiliki sumber daya terbatas.
3. Tingkat Pembelajaran *(Learning Rate)*
   FOMO: 0.001
   YOLOv5 Nano: Setelan default
   Analisis : Learning rate 0.001 pada FOMO menunjukkan bahwa model dilatih dengan tingkat pembelajaran yang relative stabil untuk menghindari osilasi atau konvergensi yang buruk. Sementara itu, pengaturan tingkat pembelajaran pada YOLOv5 Nano Setelan *default* dalam gambar 3.

2.3.5 Hasil Pelatihan Model FOMO

1 Ringkasan Kinerja Model
   F1 *Score* : Model ini memiliki F1 *Score* sebesar 99.2%, yang menunjukkan kinerja yang sangat baik dalam mengklasifikasikan data.
2. *Metric* Kebingungan
   Metric kebingungan menunjukkan bagaimana model mengklasifikasikan berbagai kategori:
   BACKGROUND : 100% akurasi dalam mengidentifikasi latar belakang.
   BOTOL : 100% akurasi dalam mengidentifikasi botol.
   MAINAN : 96.7% akurasi, dengan beberapa kesalahan.
   WAJAH: 94.1% akurasi, juga dengan beberapa kesalahan.
3. *Metric* Kinerja
   Precision (non-background): 1.00, menunjukkan bahwa semua prediksi non-latar belakang adalah benar.
   Recall (non-background): 0.98, menunjukkan bahwa model berhasil mengidentifikasi 98% dari semua contoh non-latar belakang.
   F1 Score (non-background): 0.99, menunjukkan keseimbangan yang baik antara precision dan recall.
4. Kinerja di Perangkat
   Waktu Inferensi: 1009 ms, menunjukkan waktu yang dibutuhkan model untuk membuat prediksi.
   Penggunaan RAM Puncak: 235.6 K (kilobyte) , menunjukkan penggunaan memori maksimum saat model beroperasi.
   Penggunaan *Flash*: 65.2 K (kilobyte), menunjukkan penggunaan memori *flash.*



Gambar 4 Pelatihan Model FOMO

2.3.6 Hasil Pelatihan Model YOLOv5 Nano

1 Ringkasan Kinerja Model

Precision Score: 86.9%

Ringkasan kinerja model ini menunjukkan seberapa akurat model dalam mengklasifikasikan objek terdeteksi. Semakin tinggi akurasinya, semakin baik performa model.

2 *Metric* (Set Validasi)

mAP (*Mean Average Precision):* 0.61

Ringkasan kinerja model ini adalah *metric* umum yang digunakan untuk mengevaluasi kinerja model pendeteksian objek. *metric* ini menyajikan kinerja relative dari berbagai persimpangan terhadap persimpangan (IoU).

mAP@[IoU=50]: 1.00

Model berhasil mendeteksi objek dengan akurasi sempurna pada ambang batas IoU 50%.

mAP@[IoU=75]: 0.70

Model memiliki presisi yang baik pada ambang batas IoU 75%.

mAP berdasarkan ukuran area:

mAP@[area=small]: -1.00

Nilai ini menunjukkan bahwa model tidak berhasil mendeteksi objek kecil.

mAP@[area=medium]: 0.61

Model menunjukkan kinerja yang baik dalam mendeteksi objek dengan ukuran sedang.

mAP@[area=large]: -1.00

Model juga tidak berhasil mendeteksi objek besar.

3 *Recall*

Recall@[max_detections=1] : 0.68

Recall@[max_detections=10] : 0.68

Recall@[max_detections=100] : 0.68

Ringkasan kinerja model ini menunjukkan proporsi objek yang berhasil dideteksi oleh model dibandingkan dengan total objek yang ada. Nilai yang sama pada semua ambang batas menunjukkan konsistensi dalam deteksi.

4 *Recall* berdasarkan ukuran area :

Recall@[area=small]: -1.00

Model tidak dapat mendeteksi objek kecil.

Recall@[area=medium] : 0.68

Recall@[area=large] : -1.00
Model tidak dapat mendeteksi objek besar.

5 Kinerja di Perangkat
Waktu Inferensi: 5034 ms
Ringkasan kinerja model ini adalah waktu yang dibutuhkan model untuk memproses dan memberikan hasil deteksi.
Penggunaan RAM Puncak: 220.8 K *(kilobyte)*
Ini menunjukkan jumlah memori yang digunakan oleh model saat beroperasi.
Penggunaan *Flash:* 1.8 MB *(megabyte)*
Ini menunjukkan penggunaan memori *flash* oleh model.

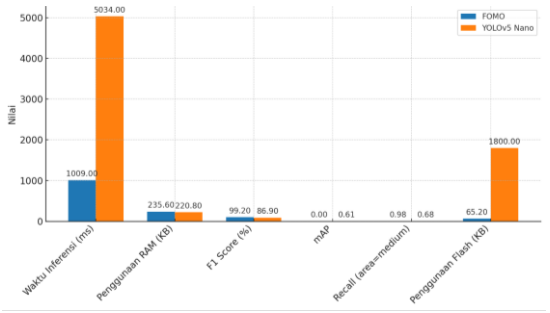
Gambar 5 Hasil Pelatihan Model YOLOv5 Nano

3. HASIL DAN PEMBAHASAN

Metric yang digunakan untuk mengevaluasi kinerja model meliputi *Precision*, *Recall*, dan F1 *Score*. *Precision* mengukur seberapa akurat model dalam mengklasifikasikan objek yang terdeteksi, sementara *Recall* mengukur seberapa banyak objek yang berhasil dideteksi dari total objek yang ada. F1 *Score* merupakan ukuran yang menggabungkan *precision* dan *recall.* Secara sederhana, mAP menunjukkan seberapa akurat dan lengkap model mendeteksi objek. Nilai mAP yang lebih tinggi menunjukkan bahwa model mendeteksi objek dengan lebih akurat dan menghindari kesalahan deteksi, sedangkan nilai mAP yang lebih rendah menunjukkan bahwa model masih memiliki banyak kesalahan dalam deteksi objek, baik dalam hal *precision* dan *recall*. Tolak ukur nilai tersebut mengukur tingkat akurasi dan ketepatan perangkat ESP32-CAM dalam menjalankan model algortima yang telah dilatih berdasarkan *environment* yang di sesuaikan dengan ESP32-CAM melalu platform *Edge Impulse* seperti di tabel 1 Hasil Penilaian Kinerja kedua algoritma.

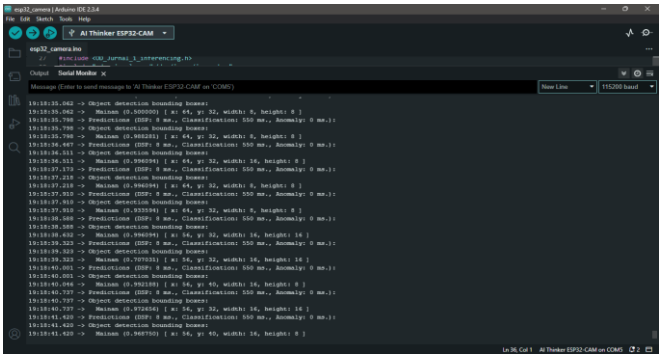
Tabel 1 Hasil Penilaian Kinerja kedua Algoritma

metric	**FOMO**	**YOLOv5 Nano**
Waktu Inferensi	1009 ms	5034 ms
Penggunaan RAM	235.6 K *(kilobyte)*	220.8 K *(kilobyte)*
F1 *Score*	99.2%	86.9%
mAP	-	0.61
Recall (area=medium)	0.98	0.68
Penggunaan *Flash*	65.2 K *(kilobyte)*	1.8 MB *(Megabytes)*

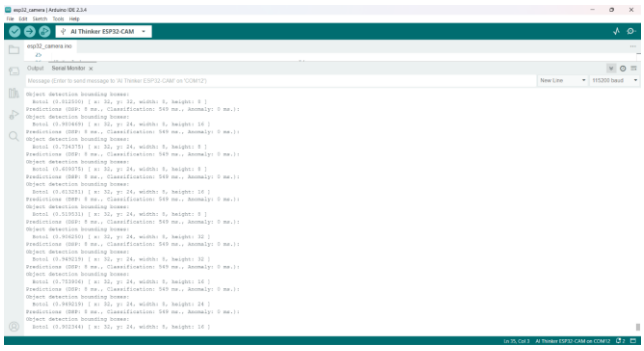


Gambar 6 Grafik Perbandingan Kinerja FOMO dan YOLOv5 Nano

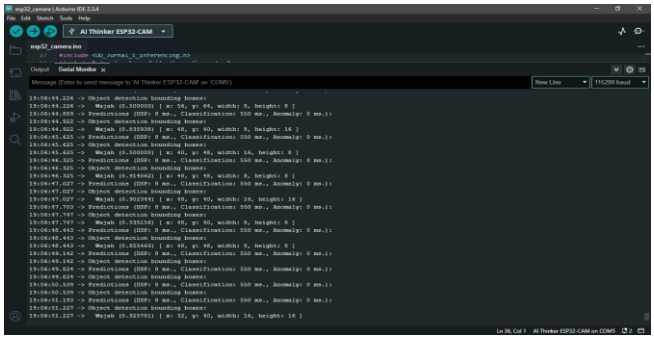
Dari hasil pengujian langsung menggunakan ESP32-CAM dengan algoritma YOLOv5 Nano dan FOMO, Gambar di bawah ini menunjukkan hasilnya.



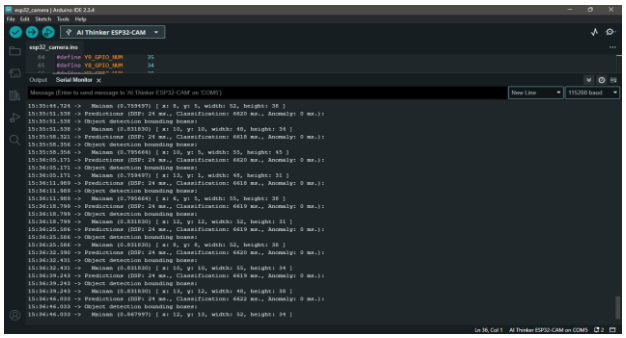
Gambar 7 Objek Mainan Algoritma FOMO



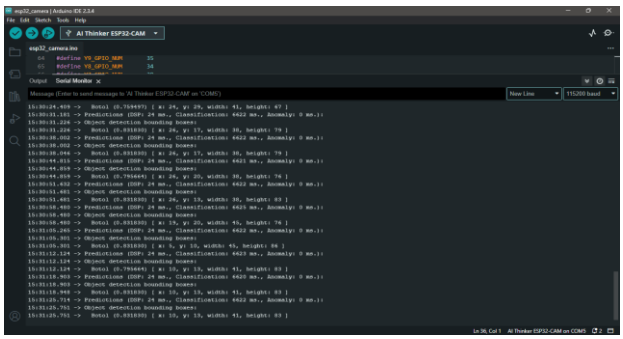
Gambar 8 Objek Botol Algoritma FOMO



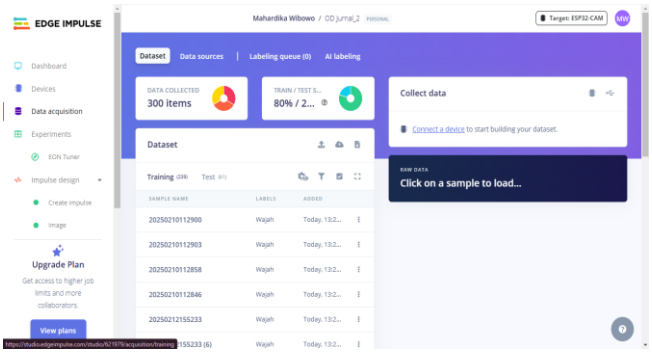
Gambar 9 Objek Wajah Algoritma FOMO



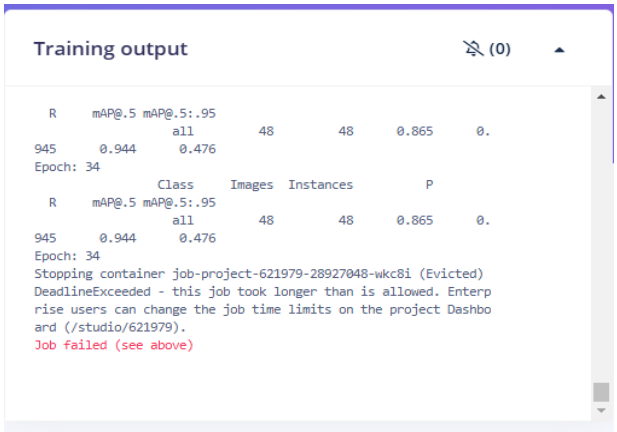
Gambar 10 Objek Mainan Algoritma YOLOv5 nano



Gambar 11 Objek Botol Algoritma YOLOv5 nano



Gambar 12 *Model Machine Learning* Objek Wajah Algoritma YOLOv5 Nano



Gambar 13 *Error* Pelatihan Objek Wajah Algoritma YOLOv5 Nano

3.1 Analisis

1. Kecepatan
   FOMO secara signifikan lebih cepat dari YOLOv5 Nano, dengan waktu inferensi sekitar lima kali lebih cepat. Ini merupakan keuntungan besar untuk aplikasi berbasis *realtime*. FOMO dapat mendeteksi lebih cepat karena penggunaan sumber daya yang lebih kecil. Dapat dilihat pada waktu interferensi dari hasil uji coba kedua algoritma tersebut pada table 1. Hanya saja FOMO memerlukan jumlah *dataset* yang lebih banyak.
2. Akurasi
   Meskipun tidak memiliki mAP untuk FOMO, F1 *Score* yang sangat tinggi (99.2%) menunjukkan bahwa model tersebut berkinerja sangat baik pada *dataset* validasi. Namun, penting untuk diingat bahwa *dataset* validasi mungkin tidak sepenuhnya representatif dari data di dunia nyata. Idealnya membutuhkan mAP untuk perbandingan yang lebih tepat dengan YOLOv5 Nano. Pada uji coba yang telah dilakukan menggunakan skema *dataset* yang lebih kecil, YOLOv5 Nano lebih baik daripada FOMO dalam tingkat akurasi.
3. Penggunaan RAM
   Penggunaan RAM FOMO sedikit lebih tinggi daripada YOLOv5 Nano, tetapi perbedaannya relatif kecil.
4. Penggunaan Flash
   FOMO menggunakan flash Memori jauh lebih sedikit daripada YOLOv5 Nano.
5. Deteksi Wajah dan Keterbatasan *Edge Impulse*
   YOLOv5 Nano menghadapi kendala dalam mendeteksi wajah akibat keterbatasan pada *platform Edge Impulse*, terutama dalam hal waktu pemrosesan yang lama. Faktor ini perlu dipertimbangkan dengan cermat, terutama jika deteksi wajah merupakan persyaratan utama. Jika YOLOv5 Nano tidak dapat menjalankannya secara efisien pada *Edge Impulse*, maka FOMO dapat menjadi alternatif yang lebih baik dengan catatan bahwa FOMO mampu melakukan deteksi wajah dengan kinerja yang memadai. Oleh karena itu, diperlukan pengujian dan evaluasi lebih lanjut untuk memastikan efektivitasnya.

4. KESIMPULAN

Penelitian ini menunjukkan bahwa FOMO memiliki keunggulan signifikan dalam hal waktu inferensi dibandingkan dengan YOLOv5 Nano, dengan waktu inferensi yang hampir lima kali lebih cepat. Meskipun F1 *Score* FOMO sangat tinggi, dengan nilai 99,2%, Namun FOMO memerlukan *dataset* yang lebih banyak dari YOLOv5 Nano. Berbanding dengan YOLOv5 Nano pada skema *dataset* yang lebih kecil tingkat akurasi terbilang lebih baik dari FOMO, hanya saja penggunaan sumber daya yang lebih besar menjadi kekurangan YOLOv5 Nano seperti yang terlihat pada tabel 1 interferensi masing – masing algrotima.

5. SARAN

Kedepannya, disarankan untuk menguji FOMO dan YOLOv5 Nano pada perangkat lain dengan sumber daya terbatas dan membandingkan kinerjanya dengan algoritma deteksi objek lainnya. Hal ini sebagai tolak ukur untuk menilai seberapa efektif FOMO dan YOLOv5 Nano dalam mengatasi keterbatasan sumber daya perangkat serta untuk mengetahui kelebihan dan kekurangan masing - masing algoritma. Optimasi lebih lanjut pada YOLOv5 Nano, terutama dalam hal waktu inferensi dapat di lakukan dengan menggunakan metode *prunning* dan skema yang berbeda.

DAFTAR PUSTAKA

[1] Saputra, D. H., Imran, B., & Juhartini. (2023). *Object detection untuk mendeteksi citra buah-buahan menggunakan metode YOLO*. Jurnal Kecerdasan Buatan dan Teknologi Informasi.

[2] Iqbal, M. N., Midyanti, D. M., & Bahri, S. (2024). *Deteksi objek manusia pada citra menggunakan single shot detector (SSD) berbasis edge computing*. Jurnal Teknologi Informasi dan Ilmu Komputer.

[3] Li, X., Fan, D., Meng, C., Zeng, L., & Yuan, T. (2022). Embedded Implementation of YOLO Nano Algorithm for On-Site Recognition of Relay Pressure Board Status. In Applied Mathematics, Modeling and Computer Simulation (pp. 9-18). IOS Press.

[4] Saputra, F. A., & Chandra, J. C. (2022). *Prototipe sistem keamanan ruang server otomatis menggunakan ESP32CAM dan algoritma You Only Look Once (YOLO)*. Jurnal Ticom: Technology of Information and Communication.

[5] Hasbullah, A. W., Setiawan, E., & Rachmad, A. (2023). *Evaluasi keandalan model rekognisi suara burung hama menggunakan platform Edge Impulse pada mikrokontroller low power*. Jurnal Teknik Elektro dan Komputer TRIAC, 10(2), 69–75

[6] Jamaludin, F., Asriyanik, A., & Pambudi, A. (2024). *Penerapan YOLO (You Only Look Once) untuk deteksi etika berbusana di Universitas Muhammadiyah Sukabumi*. JATI (Jurnal Mahasiswa Teknik Informatika).

[7] Sari, D. P., & Rahman, A. (2023). *YOLO-V8: Peningkatan algoritma untuk deteksi objek dalam kondisi pencahayaan rendah*.

[8] Yanto, Y., Aziz, F., & Irmawati, I. (2023). *YOLO-V8 peningkatan algoritma untuk deteksi pemakaian masker wajah*. JATI (Jurnal Mahasiswa Teknik Informatika).

[9] Hafidzulrahman, D. (2023). *Perbandingan algoritma You Only Look Once (YOLO) versi 5 dan versi 8 sebagai object detection pada pendeteksian hilal*. Universitas Islam Negeri Jakarta.

[10] Jalaludin, R., & Laksmiati, D. (2023). *Perancangan sistem kendali irigasi otomatis dan pengusir hama burung dengan menggunakan sensor PIR*. Jurnal Ilmiah Telsinas Elektro, Sipil dan Teknik Informasi.

[11] Badgujar, C. M., Poulose, A., & Gan, H. (2024). *Agricultural object detection with You Only Look Once (YOLO) algorithm: A bibliometric and systematic literature review*. ArXiv, abs/2401.10379.

[12] Rahman, A. A., Agustin, S. D., Ibrahim, N., & Kumalasari, N. C. (2022). *Perbandingan algoritma YOLOv4 dan Scaled YOLOv4 untuk deteksi objek pada citra termal*. MIND Journal.

[13] Kang, S., Hu, Z., Liu, L., Zhang, K., & Cao, Z. (2025). Object Detection YOLO Algorithms and Their Industrial Applications: Overview and Comparative Analysis. *Electronics*, *14*(6), 1104.

[14] Hussain, M. (2023). YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, *11*(7), 677.

[15] Gheorghe, C., Duguleana, M., Boboc, R. G., & Postelnicu, C. C. (2024). Analyzing Real-Time Object Detection with YOLO Algorithm in Automotive Applications: A Review. *CMES-Computer Modeling in Engineering & Sciences*, *141*(3).