Journal of Computer Science and Information Technology (JCSIT)
Volume 1, No 2 – Maret 2024
e-ISSN : 3031-8467

PENDETEKSIAN OBJEK MENGGUNAKAN OPENCV DAN METODE YOLOv4-TINY UNTUK MEMBANTU TUNANETRA

Randy Moh Yusup[1]*, Aldof Faris Anugrah[2], Dinda Desmonda Muslimah[3] , Sri Mentari Widya Ningrum Permana[4], Shindi Yuliani[5]
[1,2,3,4,5]Universitas Majalengka, Indonesia
[1]randymohyusup@gmail.com, [2]aldoffaris02@gmail.com , [3]dindadesmonda@gmail.com
[4]srimentari356@gmail.com, [5]shindiyuliani19@gmail.com

Received: 19-02-204	Revised: 28-02-2024	Approved: 09-03-2024

ABSTRAK

Deteksi objek adalah tugas mendasar dalam computer vision dan memiliki banyak aplikasi di berbagai bidang seperti autonomous vehicles, sistem pengawasan, dan robotika. Jurnal ini menyajikan studi komprehensif tentang deteksi objek menggunakan kombinasi dari OpenCV dan YOLOv4-Tiny, yang merupakan sebuah algoritma pembelajaran mendalam yang canggih. OpenCV adalah perpustakaan computer vision sumber terbuka secara luas yang dikenal dengan koleksi fungsi dan algoritma yang luas. Di sisi lain , YOLOv4 Tiny adalah varian ringkas dari algoritma deteksi objek YOLO (You Only Look Once), yang dirancang untuk mencapai performa waktu nyata tanpa mengurangi akurasi. Dalam studi ini, kami memanfaatkan kemampuan OpenCV dan YOLOv4-Tiny untuk mengembangkan sistem deteksi objek yang kuat. Pertama, kami memberikan tinjauan mendetail tentang arsitektur YOLOv4-Tiny, barpusat pada komponen utamanya, termasuk backbone network, feature pyramid, dan detection layers. Kesimpulannya, jurnal ini memberikan eksplorasi komprehensif tentang deteksi objek menggunakan OpenCV dan YOLOv4 Tiny. Studi tersebut menyoroti keuntungan dari kombinasi ini dalam hal kecepatan dan akurasi dan menghadirkan implementasi praktis dari sistem tersebut. Hasilnya menampilkan potensi sistem untuk aplikasi deteksi objek real-time, berkontribusi pada kemajuan visi komputer dan berbagai domainnya. Selain itu, kami mengevaluasi kinerja sistem kami pada kumpulan data tolok ukur standar, seperti COCO (Common Objects in Context), untuk menilai akurasi pendeteksian dan efisiensi komputasinya.
Kata Kunci: Deteksi objek, OpenCV, YOLOv4

PENDAHULUAN

Sistem visual manusia secara alami sangat akurat dan tepat, sehingga manusia dapat melakukan banyak hal sekaligus bahkan dalam keadaan pikiran yang kurang sadar[1]. Namun, beberapa orang dengan kondisi kebutaan tidak memiliki kemampuan ini. Kebutaan, atau juga dikenal low vision, adalah kondisi di mana seseorang memiliki penurunan kemampuan mereka untuk melihat dan memvisualisasikan dunia luar, yang mengurangi mobilitas dan produktivitas mereka dalam menyelesaikan tugas sehari-hari. Dalam jangka panjang, gangguan penglihatan berat atau kondisi kebutaan yang terjadi sejak dini dapat mengganggu perkembangan verbal, emosional, sosial, dan kognitif anak[2]. Seseorang dengan kondisi ini biasanya bergantung pada tongkat atau orang lain untuk membantu mereka berjalan ataupun menghindari rintangan, mereka juga tidak dapat menyadari perubahan mendadak dalam suatu lingkungan yang di mana tidak memungkinkan mereka untuk bereaksi secara spontan[3].

Namun, kemajuan teknologi saat ini telah sangat maju sehingga beberapa teknologi didalamnya dapat dimanfaatkan untuk membantu orang dengan kebutaan atau tunanetra. Salah satu implementasi teknologi yang dapat digunakan adalah pendeteksian objek melalui kamera ponsel, dengan sistem pendeteksian objek diharapkan dapat menemukan objek di dunia nyata dengan memanfaatkan model objek yang telah diketahui sebelumnya[4].

Deteksi objek merupakan salah satu bidang dari *computer vision* yang dapat mendeteksi objek semantik dalam gambar atau video[5]. Deteksi objek merujuk sebagai suatu metode yang berfungsi untuk menemukan dan mengidentifikasi keberadaan suatu objek dari kelas tertentu[6][7]. Fungsi adanya pendeteksi objek ialah untuk membantu mobilitas penyandang tunanetra dengan lebih mudah dalam bergerak bahkan ditempat asing sekalipun. Selain itu, mereka juga dapat lebih bebas dan mandiri untuk mengeksplorasi lingkungan luar. Tujuan pendeteksi objek adalah untuk mendeteksi semua objek yang umum dikenali di lingkup kehidupan, seperti orang, mobil, dan lainnya. Salah satu alasan mengapa pendeteksi objek diperlukan adalah untuk membantu dalam pengambilan keputusan, seperti berhenti, berbelok, dan lainnya, serta alasan utamanya yaitu untuk mengenali objek yang ada disekitar posisi pengguna berada[8]. *You Only Look Once*(*YOLO*) merupakan salah satu algoritma yang dapat digunakan untuk membangun aplikasi pendeteksi objek, dimana pengimplementasiannya disertai dengan pemanfaatan *OpenCV*.

Bagi manusia, pengenalan objek adalah proses yang sederhana, namun bagi komputer, itu merupakan hal yang sulit, karena membutuhkan tahapan proses untuk mengenali, mengidentifikasi, dan menemukan objek dengan tingkat presisi tertentu[9]. Algoritma *You Only Look Once* (*YOLO*) akan memprediksi objek apa yang ada disekitar dan dimana letak keberadaan objek tersebut berada, *YOLO* mampu mendeteksi objek dengan *frame rate* yang lebih tinggi[10]. *YOLO* termasuk salah satu metode pendeteksian objek tercepat dengan kinerja real-time yang baik dan akurasi tinggi serta memiliki struktur jaringan yang kompleks[11][12][13]. Dalam melatih gambar objek serta uji coba, metode *YOLO* melampaui metode deteksi lainnya seperti *DPM* dan *R-CNN*[14]

Program aplikasi pendeteksi objek yang dibangun menggunakan Bahasa pemograman python, serta, library OpenCv untuk fungsi pemrosesan masukan. *OpenCv* merupakan library dari fungsi pemrograman untuk *real-time computer vision* dan untuk membedakan setiap objek yang tertangkap oleh kamera[15][16]. *OpenCv* menggunakan lisensi BSD (lisensi Distribusi Perangkat Lunak Berkeley) dan bersifat *open-source* baik untuk penggunaan akademik maupun komersial [17].

Berdasarkan data dari Kementerian Kesehatan RI, jumlah penyandang disabilitas tunanetra di Indonesia adalah 1,5% dari total keseluruhan populasi penduduk di Indonesia. Jika penduduk Indonesia saat ini lebih dari 270 juta jiwa, maka jumlah penyandang tunanetra berada pada kisaran 4 juta jiwa. Sejalan dengan hal tersebut, pengembangan sistem pendeteksian objek secara *real-time* disertai dengan outputan suara diharapkan mampu membantu para penyandang tunanetra dalam menjalankan aktivitas keseharian secara lebih fleksibel dan mudah.

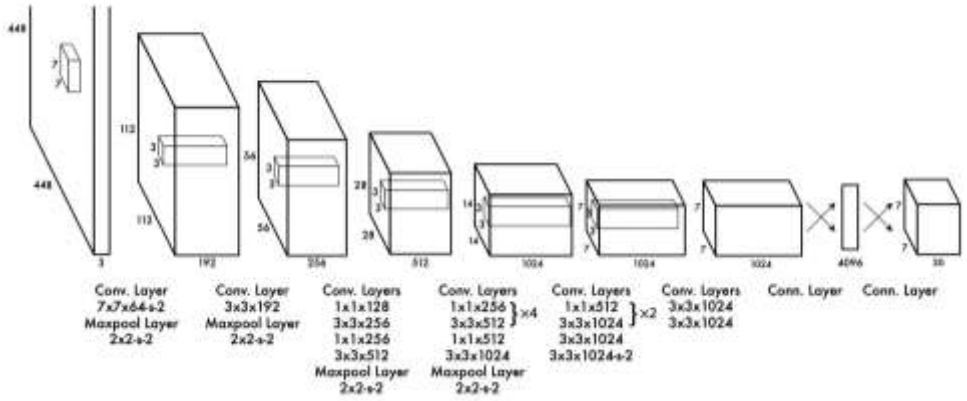
METODOLOGI PENELITIAN

Metode *YOLO* (*You Only Look Once*) ialah metode yang dapat digunakan untuk pembuatan sistem pengenalan objek. Aplikasi *YOLO* dapat diakses melalui *DARKNET* (*open source neural network)*. Cara kerja *YOLO* adalah dengan melihat gambar secara keseluruhan sebanyak satu kali, lalu melewati neural network sekali, dan kemudian mendeteksi objek yang ada, oleh karena itu dinamakan *YOLO* (*You Only Look Once*).

Arsitektur YOLO

Gambar 1. menunjukkan arsitektur *YOLO* yang dibangun atas 4 *max-pooling layer* dengan 24 lapisan konvolusi dan diikuti oleh 2 *fully connected layer*. Sebagian lapisan

konvolusi menggunakan lapisan pengurangan 1×1 sebagai opsi lain untuk pengurangan kedalaman pada peta fitur. Arsitektur *YOLO* sebenarnya relatif sederhana. Sistem menerima input citra bentuk (448,448,3) yang merupakan citra 3 channel dengan ukuran 448 x 448, yang kemudian melalui proses jaringan konvolusi untuk menghasilkan keluaran bentuk (7,7,30), dimana 7 x 7 adalah ukuran sel grid (S=7) dan 30 adalah nilai total kotak pembatas B dikalikan total kelas dan total komponen kotak B (B) x 5 +C, B =2, C=20).



Gambar 1 Arsitektur YOLO

Setiap operasi konvolusi mempunyai beberapa parameter yang mempengaruhi keluaran yang dihasilkan. Selain ukuran kernel filter dan total parameter filter, ada dua parameter lain yang menjadi faktor lain, yaitu *padding* dan *stride*. *Padding* ialah parameter yang menentukan total border yang ditambahkan ke semua tepian input. Tujuannya adalah untuk meminimalkan kehilangan informasi pada tepi citra saat menjalankan operasi konvolusi. Biasanya, kernel filter melewati tepi citra kecuali jika menggunakan kernel berukuran 1x1. *Zero-padding* adalah teknik yang umum digunakan untuk menangani masalah ini, di mana nilai 0 ditambahkan pada tepian input citra. *Stride* adalah parameter yang menentukan seberapa jauh kernel filter bergerak setiap kali melakukan konvolusi. Parameter ini sering digunakan untuk memperkecil ukuran output. Ukuran output dapat dihitung berdasarkan operasi konvolusi menggunakan persamaan[18].

Bounding Box

Dalam penelitian yang dilakukan oleh (Putra et al., 2021), dijelaskan bahwa *Bounding Box* adalah sebuah kotak imajiner yang digunakan untuk mengelilingi objek yang telah terdeteksi. *Bounding Box* memiliki bentuk kotak dan ukurannya sama dengan ukuran objek yang terdeteksi. Untuk membuat *Bounding Box*, koordinat piksel yang diperlukan adalah *upper-left* (UL), *upper-right* (UR), *lower-left* (LL), dan *lower-right* (LR).

Confidence Score

Menurut penelitian yang dilakukan oleh (Hutauruk et al., 2020), *YOLO* menggunakan *confidence score* untuk masing-masing kotak pembatas (*Bounding Box*). *Confidence score* ini merupakan probabilitas bahwa kotak tersebut berisi objek, yang dikalikan dengan *IoU* (*Intersection over Union*) antara kotak prediksi dan *ground truth* yang diperoleh selama proses pelatihan. *IoU* digunakan sebagai ukuran evaluasi untuk mengukur sejauh mana deteksi objek pada dataset yang diberikan. Perhitungan

confidence score atau nilai kepercayaan untuk setiap *bounding box* dijelaskan dalam persamaan 2 dan 3.

$$C = P_r(\text{object}).IoU_{pred}^{truth} \quad (2)$$

$$IoU_{pred}^{truth} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (3)$$

Dimana :

C = *Box Confidence Score*

$P_r(object)$ = Probabilitas kotak berisi objek, jika ada bernilai 1, jika tidak bernilai 0

IoU_{pred}^{truth} = IoU Antara kotak pada prediksi dan kebenaran dasar.

Class confidence score, yang merupakan hasil dari probabilitas kondisional kelas dan *confidence score* dari kotak pembatas, adalah faktor penentu dalam prediksi akhir. *Class confidence score* mencerminkan tingkat kepercayaan kelas secara khusus untuk setiap kotak, menunjukkan kemungkinan keberadaan kelas tertentu dalam kotak tersebut, dan sejauh mana kotak prediksi cocok dengan objek yang sebenarnya. Persamaan untuk menghitung *Class confidence score* pada setiap kotak prediksi diberikan dalam persamaan 4.

$$\boldsymbol{P_r(Class_i|object) \cdot box\ confidence\ score = P_r(Class_i) \cdot IoU_{pred}^{truth}} \quad (4)$$

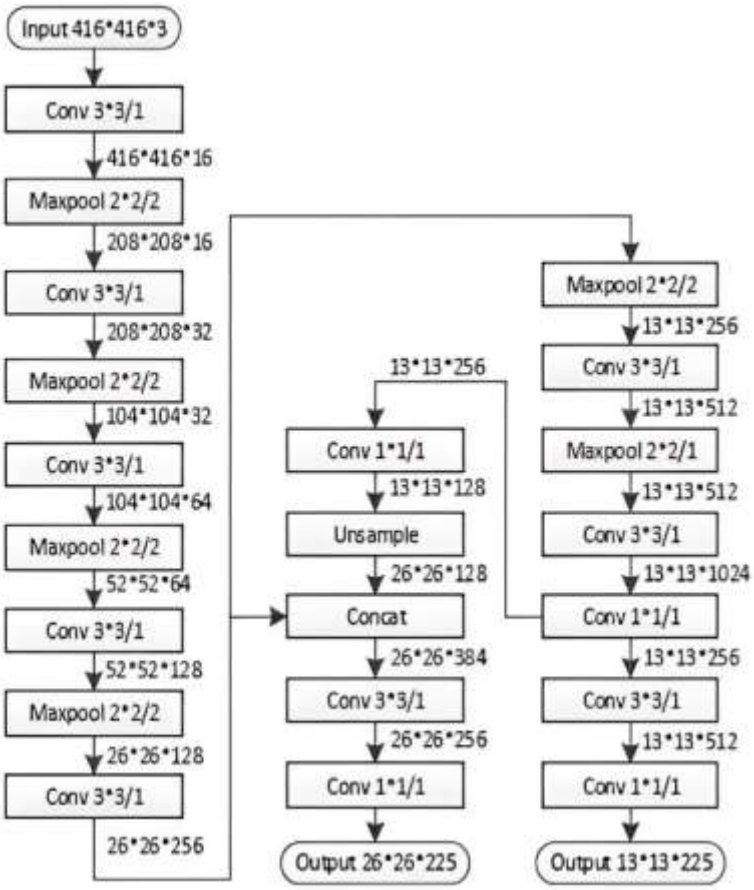
Dimana :

$P_r(Class_i|object)$ = Probabilitas kondisional kelas *i*

$P_r(Class_i)$ = Probabilitas kelas *i*

Tiny-YOLO

Dalam studi penelitian yang dilakukan oleh Adarsh et al. (2020), *Tiny-YOLO* diklasifikasikan sebagai variasi *YOLO* yang mempunyai struktur layer konvolusi yang lebih sederhana. Hal ini mengakibatkan peningkatan signifikan dalam kecepatan pengenalan sebesar 442% dibandingkan dengan versi *YOLO* sebelumnya. Namun, akurasi yang dicapai oleh *Tiny-YOLO* lebih rendah. *Tiny-YOLO* menggunakan lapisan *pooling* dan mereduksi ukuran gambar untuk lapisan konvolusi, serta melakukan prediksi kelas pada tensor 3D yang berisi skor objek, kotak pembatas, dan prediksi kelas pada dua skala yang berbeda. Gambar dibagi menjadi sel grid dengan ukuran S x S. Pada tahap deteksi akhir, *Tiny-YOLO* mengabaikan kotak pembatas yang tidak mempunyai skor objek terbaik. Ekstraksi fitur dilakukan menggunakan lapisan konvolusi dan *max-pooling* pada konfigurasi input *Tiny-YOLO.* Prediksi kotak pembatas dilakukan di dua skala peta fitur yang berbeda, yaitu 13x13 dan 26x26. Arsitektur *Tiny-YOLO* dapat dilihat pada Gambar 2.



Gambar 2 Arsitektur Tiny-YOLO

YOLOv4-Tiny

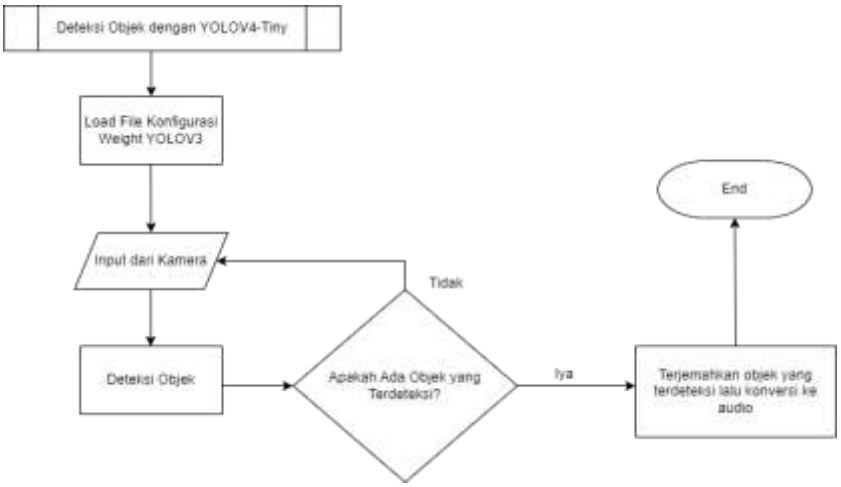
YOLOv4-tiny merupakan versi yang dikompresi dari *YOLOv4*. Pengembangan *YOLOv4-Tiny* didorong oleh kebutuhan untuk membangun struktur jaringan yang lebih sederhana dan mereduksi parameter dibandingkan dengan *YOLOv4*, sehingga cocok untuk digunakan pada *device mobile* dan *embedded*. *YOLOv4-tiny* dapat digunakan untuk pelatihan dan deteksi yang lebih cepat. *YOLOv4-tiny* memiliki dua kepala *YOLO*, sedangkan *YOLOv4* memiliki tiga kepala. *YOLOv4-tiny* dilatih dari 29 lapisan konvolusi sebelumnya, sedangkan *YOLOv4* dilatih dari 137 lapisan konvolusi sebelumnya. *Frames Per Second* (FPS) pada *YOLOv4-tiny* sekitar 8x lebih cepat dari *YOLOv4*. Namun, akurasi *YOLOv4-tiny* sebesar 2/3 dari *YOLOv4* ketika diuji pada dataset *MS COCO*. Model *YOLOv4-tiny* mencapai 22,0% AP (42,0% AP50) dengan kecepatan 443 FPS pada RTX 2080Ti. Dengan menggunakan TensorRT, ukuran batch 4, dan presisi FP16, *YOLOv4-tiny* mencapai 1774 FPS. Untuk deteksi objek secara real-time, *YOLOv4-tiny* merupakan pilihan yang lebih baik daripada *YOLOv4* dikarenakan waktu inferensi yang lebih cepat dan lebih penting dibandingkan presisi atau akurasi ketika bekerja dengan lingkungan deteksi objek real-time.

Journal of Computer Science and Information Technology (JCSIT)
Volume 1, No 2 – Maret 2024
e-ISSN : 3031-8467

Gambar 3 Waktu Inferensi

YOLOv4-Tiny dengan Feedback Audio

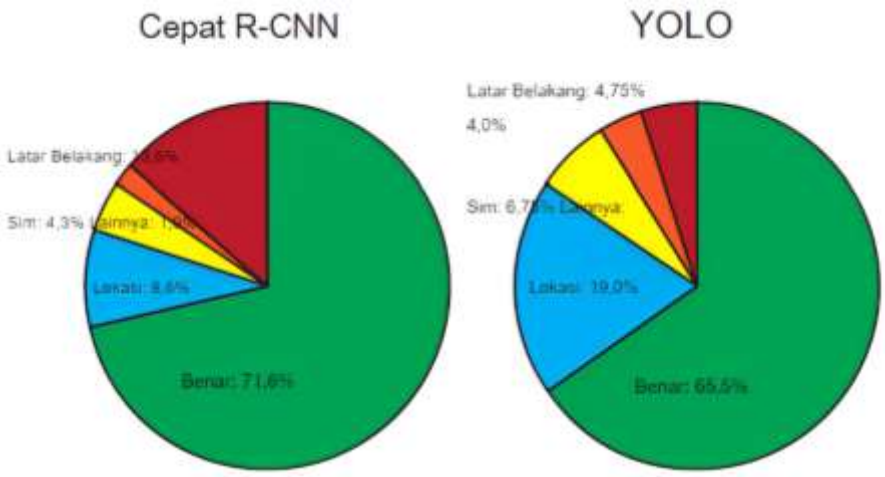
Dalam sistem yang diusulkan, sistem pendeteksian objek yang dibangun menggunakan *YOLOv4-Tiny* dengan penambahan outputan berupa audio menggunakan *gTTs* (*Google Text-To-Speech*). Dibandingkan dengan *YOLO-v4*, *FPS YOLOv4-Tiny* meningkat delapan kali lipat. Selain penggunaan *YOLOv4-Tiny*, sistem yang dibangun juga menggunakan Python3 dan *YOLO-v4-tiny weights* pra-pelatihan yang dilatih dengan modul *OpenCV Deep Neural Network (DNN)* pada dataset *COCO* (kumpulan data deteksi objek yang berisi 80 kelas dengan gambar dari pemandangan sehari-hari). Kemudian untuk mengambil masukan dari kamera dan mulai mengambil gambar, sistem menggunakan modul *OpenCV* yang sama. Frame kemudian dikirim satu per satu ke algoritma untuk mengidentifikasi objek. Posisi spasial dari objek yang diidentifikasi ditentukan. Output teks kemudian diterjemahkan ke dalam bahasa lokal menggunakan mesin terjemahan dan kemudian menjadi segmen audio menggunakan modul *gTTS*(*Google Text-To-Speech*). Umpan balik berupa suara adalah outputan utama dari sistem ini, dimana akan memberitahukan nama objek yang terdeteksi.



Gambar 4 Flowchart Alur Pendeteksi Objek

Keterbatasan YOLO

YOLO menggunakan batasan spasial yang kuat untuk memprediksi kotak pembatas, sehingga setiap sel *grid* hanya dapat memprediksi dua kotak dengan satu kelas. Batas spasial ini membatasi total objek terdekat yang dapat diprediksi oleh model. Model *YOLO* menghadapi tantangan saat berhadapan dengan objek kecil yang muncul berkelompok, seperti contohnya kawanan burung. Karena model *YOLO* belajar memprediksi *jump box* berdasarkan data, model *YOLO* sulit digeneralisasikan ke objek dengan rasio aspek atau konfigurasi baru atau tidak biasa. Selain itu, karena lapisan *down-sampling* dalam arsitektur gambar input, model *YOLO* mengandalkan fitur yang cenderung kasar untuk memprediksi *bounding box*. Saat fungsi kerugian dilatih untuk memperkirakan kinerja deteksi, fungsi kerugian menangani kesalahan dalam kotak pembatas kecil dan besar dengan cara yang sama. Kesalahan kecil dalam kotak besar biasanya tidak berbahaya, tetapi kotak kecil memiliki dampak yang jauh lebih besar pada metrik *Intersection over Union* (*IoU*) (Redmon & Divvala, 2016).



Gambar 5 Analisis Kesalahan(Redmon , Divvala, 2016)

Dalam perbandingan antara *Fast R-CNN* dan *YOLO*, Gambar 5 menunjukkan persentase kesalahan dalam menempatkan objek dengan tepat dan dalam mengidentifikasi latar belakang pada tingkat deteksi tertentu untuk berbagai kategori. Setiap kategori memiliki jumlah objek yang berbeda (N = jumlah objek dalam kategori tersebut). Kesalahan penempatan objek terjadi ketika objek salah diidentifikasi dengan IoU>0,1, sedangkan kesalahan latar belakang terjadi ketika tidak ada objek yang diidentifikasi dengan IoU<1. Angka 4 memberikan detail untuk setiap jenis kesalahan yang dihitung sebagai rata-rata dari 20 kelas. *YOLO* menghadapi kesulitan dalam menempatkan objek secara akurat. Jumlah kesalahan penempatan objek oleh *YOLO* lebih besar daripada gabungan semua metode deteksi lainnya. Di sisi lain, *Fast R-CNN* membuat lebih sedikit kesalahan penempatan objek, tetapi memiliki jumlah kesalahan latar belakang yang jauh lebih tinggi. Sekitar 13,6% dari deteksi atasnya adalah positif palsu yang tidak terdapat objek apapun. *Fast R-CNN* hampir tiga kali lebih cenderung memprediksi latar belakang daripada *YOLO* (Redmon & Divvala, 2016).

HASIL DAN PEMBAHASAN

Performa *YOLOv4-Tiny* dapat dievaluasi menggunakan beberapa metrik, termasuk rata-rata presisi rata-rata (mAP), presisi, daya ingat, dan kecepatan inferensi. Metrik ini memberikan wawasan tentang akurasi model, kemampuan mendeteksi objek, dan performa waktu nyata.

Akurasi

YOLOv4-Tiny mencapai akurasi yang cukup baik, meskipun lebih rendah dibandingkan dengan model *YOLOv4* penuh. Skor mAP, yang mengukur presisi rata-rata di berbagai kategori objek, kisaran 20-30%. Meskipun ini mungkin lebih rendah daripada model canggih lainnya, keunggulan *YOLOv4-Tiny* terletak pada kecepatan inferensi yang lebih cepat.

Gambar 6 Akurasi Objek Terdeteksi

Deteksi Objek

YOLOv4-Tiny mampu mendeteksi berbagai objek dengan presisi dan daya ingat yang baik. Itu dapat secara akurat mendeteksi objek di berbagai kategori, termasuk orang, kendaraan, hewan, dan objek sehari-hari. Kemampuan model untuk mendeteksi benda-benda kecil mungkin sedikit terganggu karena arsitekturnya yang diperkecil, tetapi kinerjanya masih cukup baik.

Gambar 7 Uji Coba Sistem Pendeteksi Objek

Kecepatan Inferensi

Salah satu keunggulan utama *YOLOv4-Tiny* adalah kecepatan inferensinya yang cepat. Model ini dirancang untuk melakukan deteksi objek secara *real-time* pada perangkat dengan sumber daya terbatas, seperti sistem tertanam atau perangkat *edge*. Ini mencapai waktu inferensi yang jauh lebih cepat dibandingkan dengan model *YOLOv4* lengkap, memungkinkan pemrosesan streaming video atau umpan kamera langsung secara *real-time*.

KESIMPULAN

Sistem aplikasi yang dibangun memiliki antarmuka yang sederhana sehingga memudahkan pengguna tunanetra dalam penggunaan aplikasi. Setelah aplikasi dibuka, kamera akan mulai merekam video secara *real-time*. lalu untuk merubah objek yang terdeteksi menjadi audio, pengguna diharuskan menekan tombol yang ada, selanjutnya sistem akan mulai memprosesnya dengan mengkonversi label objek menjadi outputan audio. Outputan tersebut dapat dihentikan dengan menekan tombol yang sama. Aplikasi ini memerlukan koneksi internet untuk menjalankannya. Akurasi maksimum pendeteksi objek mencapai 30% .

Pendeteksi objek ini bekerja secara akurat dalam mendeteksi objek yang berbeda, namun mungkin tidak berfungsi dengan baik jika objek terlalu dekat dengan kamera atau bukan bagian dari kumpulan data yang dipilih. Objek tidak boleh terlalu dekat dengan bingkai kamera dan harus ditempatkan pada jarak yang lebih jauh. Kumpulan data bobot *YOLO* yang digunakan untuk aplikasi ini hanya dilatih untuk 80 jenis objek.

Sistem yang telah dikembangkan saat ini dapat ditingkatkan lebih lanjut untuk memberi tahukan posisi pasti objek, jadi tidak hanya menghasilkan outputan berupa nama objek, namun disertai letak pasti posisi objek tersebut berada. Tingkat akurasi pendeteksi objek saat ini masih dapat ditingkatkan lagi guna meningkatkan kepercayaan pengguna terhadap keakuratan objek-objek yang dideteksi. Tingkat akurasi sistem deteksi dalam kondisi cahaya redup atau gelap perlu ditingkatkan, agar aplikasi dapat digunakan dalam kondisi apapun dan dimanapun tidak terkendala oleh faktor-faktor tersebut. Jarak antara objek dari kamera menjadi fitur yang dapat diimplementasikan pada tahap pengembangan berikutnya.

DAFTAR PUSTAKA

[1] Mahendru, Mansi, and Sanjay Kumar Dubey. "Real time object detection with audio feedback using Yolo vs. Yolo_v3." 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 2021.

[2] Guravaiah, K., Bhavadeesh, Y.S., Shwejan, P., Vardhan, A.H. and Lavanya, S., 2023. Third Eye: Object Recognition and Speech Generation for Visually Impaired. *Procedia Computer Science*, *218*, pp.1144-1155.

[3] Vaidya, Sunit, et al. "Real-time object detection for visually challenged people." 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2020.

[4] Riyadi, Agung Slamet, et al. "Perbandingan Metode ResNet, YoloV3, dan TinyYoloV3 pada Deteksi Citra dengan Pemrograman Python." (2021).

[5] Sarmah, Ankur Jyoti, et al. "Object detection and conversion of text to speech for visually impaired." *ADBU Journal of Engineering Technology* 12.2 (2023).

[6] Jiang, Chenchen, et al. "Object detection from UAV thermal infrared images and videos using YOLO models." International Journal of Applied Earth Observation and Geoinformation 112 (2022): 102912.

[7] Saxena, Meghna Raj, et al. "Real-time object detection using machine learning and opencv." Int J Inform Sci Appl (IJISA) 11.1 (2019): 0974-225.

[8] Tirpude, Parag, et al. "Real time object detection using OpenCV-Python." Int Res J Modernization Eng Technol Sci 4.5 (2022): 1-6.

[9] Joshi, Neeraj, Shubham Maurya, and Sarika Jain. "Real-time object detection and identification for visually challenged people using mobile platform." *CEUR-2823* (2021): 55-62.

[10] Hammam, H, Asyhar, A,. Wibowo, S. A,. Budiman, G.(2020). "implementation and performance analisys of You Only Look Once method As porn Censorship in video". 7(2),3631_3638.

[11] Fang, Wei, Lin Wang, and Peiming Ren. "Tinier-YOLO: A real-time object detection method for constrained environments." IEEE Access 8 (2019): 1935-1944.

[12] Diwan, Tausif, G. Anirudh, and Jitendra V. Tembhurne. "Object detection using YOLO: Challenges, architectural successors, datasets and applications." Multimedia Tools and Applications 82.6 (2023): 9243-9275.

[13] Jiang, Zicong, et al. "Real-time object detection method based on improved YOLOv4-tiny."(2020).

[14] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[15] I. corporation, "OpenCv," Itseez, 2000. [online]. **http://opencv.org.**

[16] Mittal, Naman, Akarsh Vaidya, and Shreya Kapoor. "Object detection and classification using Yolo." Int. J. Sci. Res. Eng. Trends 5.2 (2019).

[17] I. corporation, "http://opencv.org/platforms/," Itseez, 2000. [online].

[18] Shinde, S., Kothari, A., dan Gupta, V., 2018, "YOLO based Human Action Recognition and Localization", Procedia Computer Science, Vol. 133, Hal. 831-838, https://doi.org/10.1016/j.procs.2018.07.112