Jurnal Teknik Informatika (JUTIF)
Vol. 4, No. 4, August 2023, pp. 807-818

DOI: https://doi.org/10.52436/1.jutif.2023.4.4.791
p-ISSN: 2723-3863
e-ISSN: 2723-3871

DESIGN AND BUILD VEHICLE PLATE DETECTION SYSTEM USING YOU ONLY LOOK ONCE METHOD BASED ON ANDROID

Yolan Anjani Usen[1],Cynthia Hayat[2]

[1,2]Information Systems, Faculty of Engineering and Computer Science, Universitas Kristen Krida Wacana, Indonesia
Email: [1]yolan.2018si013@civitas.ukrida.ac.id, [2]cynthia.hayat@ukrida.ac.id

(Article Received: Desember 26, 2022; Revision: January 19, 2023; Published: August 18, 2023)

Abstract

The method of collecting the vehicle data is conducted conventionally by gathering data from each region to be converted into single, raw information in the form of vehicle plates for all regions, to be processed on a computer and sent to the Central Bureau of Statistics. It is then transformed into a form of national data file that provides information on vehicle plates for the Indonesian people. This kind of data gathering method requires a lot of time and effort. Therefore, it is a concern for researchers to detect vehicle plates using image processing by utilizing the Android-based You Only Look Once method. The YOLOv4 technique is used because it processes image data directly with optimal performance in order to produce faster predictions. In its application, the researchers use Google Collaboratory to create models and Android Studio for android applications. At the same time, the parameters studied were precision, recall, F1 score, average IoU, and mAP. By using the "Vehicle Registration Plate" dataset, the ratio of which is 70% in training data and 30% in data validation, an accuracy of 77% is obtained with a detection time of 0.05 seconds, whereas the average accuracy value is 86.82%. Therefore, it can be concluded that this study has an optimized performance for detecting vehicle plates using the Android application.

Keywords: *image processing, license plate, object detection, YOLO*

RANCANG BANGUN SISTEM DETEKSI PLAT KENDARAAN MENGGUNAKAN METODE YOU ONLY LOOK ONCE BERBASIS ANDROID

Abstrak

Data kendaraan yang dikumpulkan pada periode tertentu seringkali digunakan untuk mengetahui tingkat pertumbuhan kendaraan. Pada proses pengumpulan data kendaraan dilakukan secara konvensional dengan mengumpulkan data dari setiap daerah untuk dikumpulkan menjadi satu data mentah berupa plat kendaraan seluruh daerah untuk kemudian diolah dalam komputer dan mengirimkannya ke Badan Pusat Statistik sehingga menjadi berkas informasi berupa data nasional yang menyediakan informasi plat kendaraan masyarakat Indonesia. Dengan menggunakan cara tersebut cukup membutuhkan banyak waktu yang panjang. Sehingga menjadi perhatian untuk melakukan penelitian yang dapat mendeteksi plat kendaraan menggunakan pengolahan citra dengan memanfaatkan metode *You Only Look Once* berbasis android. Penggunaan metode YOLOv4 digunakan karena melatih data gambar secara langsung dengan kinerja yang optimal agar dapat menghasilkan prediksi yang lebih cepat. Kemudian dalam penerapannya menggunakan *Google Collaboratory* untuk membuat model dan *Android Studio* untuk aplikasi *android*. Dengan menggunakan dataset *"Vehicle Registration Plate"* yang rasionya 70% pada data *training* dan 30% pada data *validation* didapatkan hasil akurasi pada pengujian gambar sebesar 77% dengan waktu deteksi 0.05 detik yang mana nilai akurasi rata-ratanya sebesar 86.82% dan parameter yang diperhatikan adalah nilai presisi 0.85, *recall 0.83*, F1 *score 0.84*, rata-rata IoU 65.46%, dan mAP sebesar 86.82%.. Sehingga dapat disimpulkan sistem deteksi plat kendaraan ini mempunyai performa yang sangat baik untuk mendeteksi plat kendaraan menggunakan aplikasi *android*.

Kata kunci: *deteksi objek, pengolahan citra, plat kendaraan, YOLO.*

1. PENDAHULUAN

Pertumbuhan penduduk seringkali diikuti dengan pertumbuhan kendaraan dikarenakan meningkatnya mobilitas yang dilakukan penduduk untuk menunjang kegiatan sehari-hari, walaupun sudah banyak transportasi umum yang dapat digunakan namun masih banyak masyarakat yang

memilih menggunakan kendaraan pribadi sehingga menambah kepadatan kendaraan bermotor.

Menurut data yang diperoleh dari Badan Pusat Statistik tahun 2021 jumlah kendaraan bermotor yang berada di Indonesia terdapat sebanyak 143 juta unit kendaraan bermotor. Dengan daerah Jawa Timur menempati posisi pertama yang mempunyai jumlah kendaraan bermotor paling banyak di Indonesia dan disusul oleh DKI Jakarta.[1]

Disisi lain kepadatan kendaraan bermotor juga dapat menyebabkan masalah yang sudah umum terjadi di lalu lintas yaitu kemacetan. Kemacetan yang sering terjadi disebabkan oleh masalah sarana dan prasarana transportasi umum yang kurang memadai sehingga masyarakat banyak yang beralih menggunakan transportasi pribadi. Meningkatnya penggunaan kendaraan pribadi juga selain dikarenakan kenyamanan selain itu dikarenakan rendahnya biaya penggunaan kendaraan seperti sepeda motor.

Ditambah dengan meningkatnya kepadatan kendaraan maka pemilik kendaraan juga wajib untuk mendaftarkan kendaraannya ke pihak berwenang agar memiliki tanda pengenal kendaraannya seperti plat kendaraan. Dikarenakan setiap plat kendaraan bermotor yang mempunyai keunikan tersendiri untuk setiap kodenya, hal ini bertujuan untuk memudahkan pihak berwenang dalam mengenali pemilik kendaraan dari plat kendaraannya.[2]

Karena banyaknya jenis kendaraan yang berada di pasaran yang memiliki kemiripan merek hingga jenis sehingga membutuhkan bagian lain pada kendaraan yang membedakan kendaraan lainnya yaitu plat kendaraan. Umumnya proses pengumpulan data kendaraan oleh Badan Pusat Statistik dilakukan secara manual dengan mengumpulkan data dari setiap daerah kemudian memasukkan data mentah dari setiap daerah ke komputer dan mengirimkannya ke Badan Pusat Statistik pusat menjadi kumpulan data nasional berupa data kendaraan masyarakat Indonesia.[3] Namun pengumpulan data dengan cara tersebut dinilai kurang efektif dan efisien karena proses pengambilan datanya yang memakan waktu lama dari tahap pengumpulan data di setiap daerah sampai data terkumpul di pusat untuk kemudian diolah menjadi data nasional sehingga dibutuhkan rancangan sistem baru yang dapat membantu dalam mengumpulkan data kendaraan menjadi lebih efektif dan efisien. Oleh karena itu dibutuhkan rancangan sistem baru yang terdigitalisasi yang dapat membantu proses pengenalan plat kendaraan menjadi lebih mudah bagi institusi terkait. Dengan menggunakan *image processing* yaitu dengan melakukan deteksi objek yang pada penelitian ini menggunakan algoritma YOLO sehingga dapat membantu dalam mempermudah proses mendeteksi plat kendaraan menjadi lebih efektif dan efisien.

Metode *You Only Look Once* versi 4 atau biasa disingkat YOLOv4. Metode YOLO merupakan salah satu metode deteksi objek yang dapat mendeteksi objek gambar pada kondisi *real time* dengan tingkat akurasi yang tinggi dan waktu deteksi yang cepat[5][6]. Apabila dibandingkan dengan teknologi *image processing* lainnya seperti R-CNN yang memanfaatkan *Convolutional Neural Network* (CNN) yang bekerja dengan memanfaatkan kotak pembatas untuk mengklasifikasi gambar sedangkan YOLO merupakan bagian dari CNN cerdas yang dapat melakukan pengenalan objek dengan lebih cepat dengan mengenali seluruh gambar selama pelatihan menggunakan *unified model* sehingga pada lapisan s*ingle convolutional network* memprediksi banyak kotak pembatas dan juga probabilitas yang dihasilkan secara bersamaan[6][7]. Algoritma YOLO menerapkan CNN yang dalam proses pelatihan gambar algoritma YOLO dapat membagi gambar ke dalam kotak pembatas yang akan diprediksi dan menunjukan probabilitas label untuk menandai *bounding box* sesuai dengan labelnya. YOLO juga melatih data gambar secara langsung dengan kinerja yang optimal agar dapat menghasilkan prediksi yang lebih cepat.

Penelitian terdahulu mengenai pengenalan objek menggunakan metode YOLO sudah banyak dilakukan. Salah satu penelitian yang menggunakan metode YOLO untuk pendeteksian objek adalah penelitian mengenai *People Counting* untuk transportasi publik[4]. Penelitian ini menggunakan algoritma YOLOv4 dan mengambil data dari *Open Images Dataset* dengan kelas *human face*. Metode ini menghasilkan akurasi terbaik sebesar 72.68% pada skenario *training* data. Sementara pada pengujian data dalam penelitian ini hanya menilai akurasi rata-rata sistem dalam mendeteksi dan menghitung orang dengan *pre-trained weights* oleh COCO dataset adalah 62% dengan akurasi terbaik sebesar 90% namun kekurangannya adalah sulit mendeteksi orang yang menggunakan aksesoris dan duduk berdempetan.

Setelah itu, penelitian yang membahas deteksi plat kendaraan sudah pernah dilakukan sebelumnya oleh beberapa penelitian terdahulu, salah satunya penelitian yang menjadi inspirasi dari penelitian ini adalah "Penerapan Metode YOLO dan *Tesseract-OCR* Untuk Pendataan Plat Nomor Kendaraan Bermotor Umum di Indonesia Menggunakan *Raspberry Pi*"[8] Penelitian ini menggunakan metode YOLO untuk mengenali plat kendaraan dengan menggunakan *Tesseract-OCR* dan *Raspberry Pi* yang membantu pengembangan teknologi dengan biaya yang lebih terjangkau.

Selanjutnya penerapan YOLO dalam penelitian lainnya adalah implementasi sistem *social distancing* dengan judul "*Social Distancing Detection Finding Optimal Angel With YOLOv3 Deep Learning Method*"[9] yang membuktikan bahwa penerapan YOLO dapat digunakan untuk

mendeteksi jarak antar orang dan risiko yang dihasilkan.

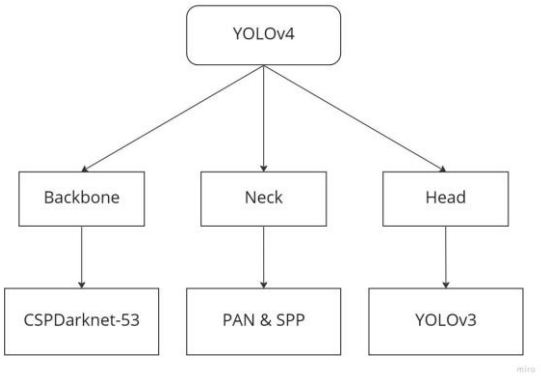
Oleh karena itu, pada penelitian ini mengusulkan penggunaan metode YOLOv4 yang memiliki performa yang lebih tinggi dari metode terdahulunya untuk mengetahui kecepatan waktu deteksi dan menghasilkan akurasi terbaik menggunakan metode YOLOv4 pada aplikasi android dalam mendeteksi plat kendaraan.

2. METODE PENELITIAN

2.1. YOLOv4

YOLOv4 merupakan algoritma deteksi objek berbasis *real time* yang menggabungkan YOLOv1, YOLOv2, YOLOv3 dan merupakan yang paling optimal untuk saat ini dalam hal kecepatan deteksi dan akurasi. pada YOLOv4 mempunyai struktur model yang terdiri dari 3 bagian yaitu: *Backbone*, *Neck*, *Prediction.*[10]

Sama seperti YOLOv3 yang masih menggunakan arsitektur *Darknet* yang menghasilkan masukan citra yang terdiri dari 53 lapisan *convolutional* sebagai *feature extraction* dan 53 lapisan lain untuk mendeteksi objek[4]. YOLOv4 juga masih menggunakan arsitektur *Darknet* tepatnya *CSPDarknet-53* sebagai *backbone* yang dapat meningkatkan *learning capability* dari CNN[4], menyambungkan modul SPP dan PAN sebagai *Neck* yang mana SPP menggantikan *pooling layer* terakhir dengan *Spatial Pyramid Pooling Layer* serta PAN untuk mendeteksi objek pada skala yang berbeda, dan YOLOv3 sebagai *Head*.[10][11]



Gambar 1. Struktur Model YOLOv4

2.2. File Konfigurasi

2.2.1 Batch Size

Batch size merupakan variabel yang menentukan jumlah banyaknya gambar atau training data yang dimasukkan saat *training* dalam satu iterasi. Selama proses pelatihan ukuran *batch* sangat mempengaruhi waktu komputasi yang diperlukan. *Batch size* juga merupakan *hyperparameter* yang penting dalam pemrosesan *deep learning*[12].

2.2.2 Subdivisions

Subdivisions merupakan hasil *batch* yang dibagikan dengan nilai subdivisi menjadi *batch* mini[13]. Kemudian *batch* mini akan dikirim ke GPU untuk proses komputasi. Selanjutnya pada proses komputasi akan diulang sampai *batch* mini selesai. Barulah proses iterasi baru dimulai sebanyak hasil pembagian antara nilai subdivisi dan *batch size*. Gambar yang lebih banyak selama proses iterasi akan membantu proses pelatihan menjadi lebih cepat, namun dapat menjadi masalah ketika subdivisi berkurang karena ada masalah pada memori yang tidak mempunyai kapasitas besar pada GPU untuk memproses lebih banyak gambar sekaligus.

2.2.3 Max Batches

Bagian ini menjelaskan banyaknya iterasi yang dilakukan pada tahap pelatihan data. sehingga semakin tingginya nilai *max batch* maka membantu sistem untuk mempelajari data lebih banyak pada data *training* yang mana nilai *max batches* harus disesuaikan dengan jumlah dari kelas label yang akan digunakan.

$$Max\ batches = number\ of\ classes * 2000 \quad (1)$$

2.3. Parameter Hasil

2.3.1 Presisi

Presisi adalah perbandingan jumlah objek yang dideteksi dengan benar dibandingkan dengan jumlah seluruh data yang diprediksi positif.[4]

2.3.2 Recall

Recall adalah perbandingan dari jumlah objek yang terdeteksi *True Positive* dibandingkan dengan seluruh data yang positif. Nilai *recall* yang tinggi menunjukkan bahwa sistem dapat mengklasifikasikan kelas objek dengan benar[4].

2.3.3 F1 Score

F1 *Score* merupakan rasio rata-rata dari nilai presisi dan *recall*. F1 *Score* memiliki nilai tertinggi sebesar 1 dan terendah sebesar 0. Nilai F1 *Score* yang semakin mendekati 1, menunjukkan bahwa kinerja sistem telah baik[4].

2.3.4 Rata-rata IoU

IoU merupakan metrik yang mengevaluasi keakuratan sistem dalam mendeteksi objek pada dataset yang telah dilatih[14] IoU membandingkan *ground-truth* atau objek pada citra dengan *predicted bounding box* dari model.

Nilai IoU pada deteksi objek berperan sebagai nilai *threshold*. Terdapat dua nilai *threshold* IoU yang umum digunakan, yaitu 0.5 dan 0.75. Pada penelitian ini, nilai *threshold* yang digunakan adalah 0.5. Jika nilai *threshold* IoU < 0.5, maka objek terdeteksi sebagai *False Positive* (FP).

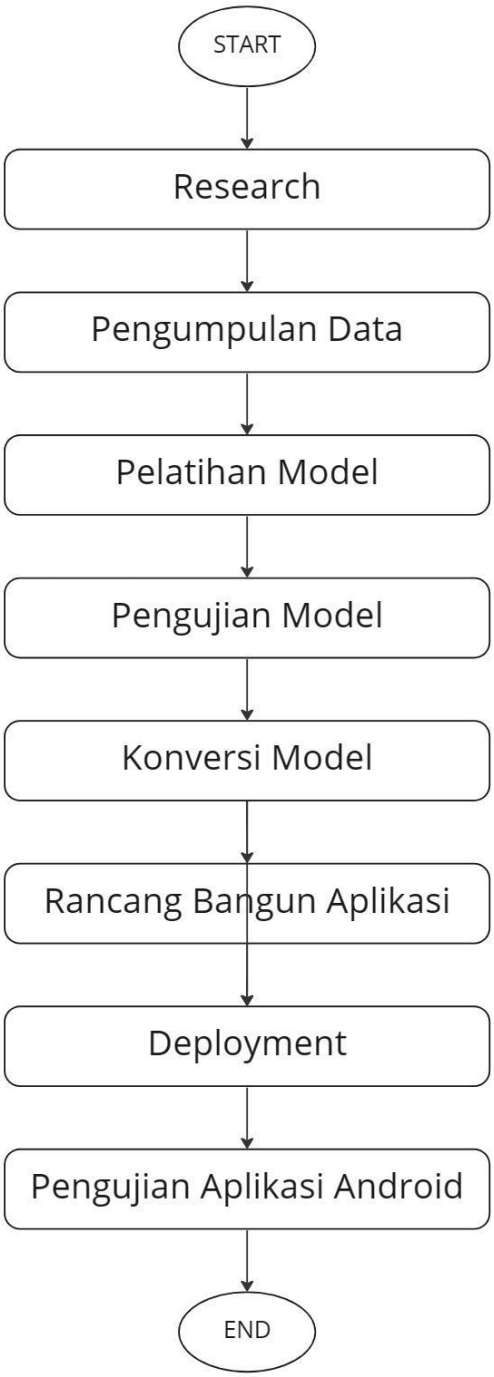
2.3.5 Mean Average Precision (mAP)

Nilai mAP merupakan rata-rata dari nilai *average precision* (AP) yang mengukur kualitas performansi dari *weights file* yang dihasilkan pada proses *training* data. Sebelum mengkalkulasikan

mAP, perlu dilakukan penyesuaian nilai *threshold* pada IoU untuk menentukan validasi dari objek yang dideteksi. Penelitian ini menggunakan nilai *threshold* IoU sebesar 0.5, sehingga model menghasilkan mAP50 yang dapat juga disebut mAP@0.5.[4]

2.4. Tahap Penelitian

Selanjutnya pada tahap penelitian dilakukan beberapa tahap dari landasan teori sampai penelitian selesai dilakukan. Maka disusunlah tahapan penelitian sebagai berikut :



Gambar 2. Alur Tahap Penelitian

Step I : Pada tahap awal penelitian, peneliti melakukan *literature research* untuk mempelajari lebih lanjut mengenai YOLOv4 yang akan digunakan untuk mendeteksi objek penelitian, kemudian mempelajari subjek yang akan diteliti yaitu plat kendaraan.

Step II : Tahap selanjutnya adalah pengumpulan data. Pada pengumpulan data akan dikumpulkan menjadi dataset berupa gambar plat kendaraan. Berikut Gambar 2 merupakan sample dari dataset plat kendaraan yang digunakan pada penelitian ini.

Gambar 3. Plat Kendaraan (Google Open Images)

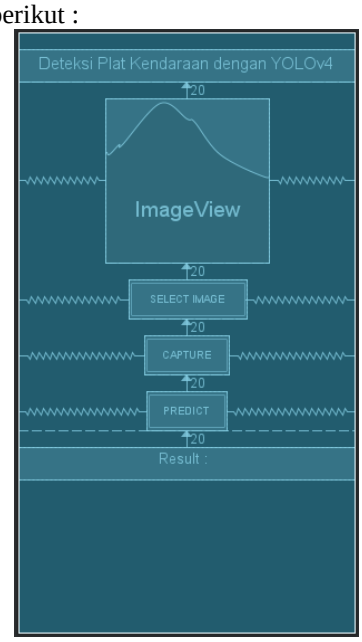
Step III : Tahap selanjutnya yaitu melatih model dengan membuat deteksi objek menggunakan *Google Collaboratory*. Proses *training* membutuhkan waktu yang cukup lama tergantung pada nilai *batch size* nya. Setiap 1000x iterasi akan menghasilkan *weight file* yang tersimpan di *storage*. Pada penelitian ini *storage* yang digunakan adalah *Google Drive*.

Step IV : Setelah proses *transfer learning* selesai maka dilakukan pengujian model yang sudah dilatih untuk memastikan model bekerja sesuai dengan dataset yang digunakan. Cara pengujiannya adalah dengan menguji *file weights* untuk mendeteksi plat kendaraan dari gambar.

Step V : *File weights* yang dihasilkan harus dikonversi menjadi format *TensorFlow Lite* agar dapat dihubungkan ke aplikasi *android* dengan mengubahnya menjadi file *protobuf* (.pb) milik *TensorFlow* kemudian konversi menjadi format *TensorFlow Lite* (.tflite). Kemudian menguji model *TFLite* untuk mendeteksi gambar.[15]

Step VI : Selanjutnya rancang fitur yang akan ada di aplikasi. Perancangan aplikasi *android* menggunakan metode *Waterfall*. Proses *deployment* dilakukan dengan cara menambah file *TFLite* ke pengkodean aplikasi. Selanjutnya pengujian aplikasi dilakukan dengan teknik *black box testing*. Adapun

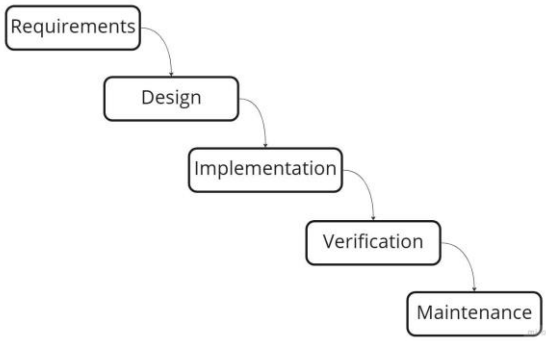
prototype aplikasinya ditampilkan pada Gambar 4 sebagai berikut :



Gambar 4. Prototype Aplikasi

2.5. Perancangan Aplikasi

Dalam perancangan aplikasi sistem deteksi plat kendaraan ini menggunakan metode *Waterfall* yang secara garis besarnya pada Gambar 5 menggambarkan langkah yang dilakukan saat merancang sistem deteksi plat kendaraan sebagai berikut:



Gambar 5. Metode Perancangan Aplikasi

Step I : Tahap pertama diawali dengan *Requirements* yang berfungsi untuk mengidentifikasi kebutuhan yang diperlukan *user* untuk membangun sistem. Dengan menggunakan teknologi pengolahan citra dibutuhkan sistem yang dapat mempermudah proses pendeteksian plat kendaraan. Dengan berbagi cara untuk mendapatkan gambar membuat sistem ini harus dapat mengakses gambar dari galeri *user* dan mengambil objek foto secara langsung melalui kamera.

Step II : Setelah kebutuhan *user* teridentifikasi, langkah selanjutnya adalah *Design* untuk merancang apa saja yang sudah dijelaskan pada langkah *Requirements* dilakukan dengan menggunakan *Use Case Diagram* dan *Activity Diagram*.

Step III : Selanjutnya merupakan tahap *Implementation* yaitu melaksanakan pengkodean program menggunakan perangkat lunak pendukung.

Step IV : Kemudian juga dilakukan tahap *Verification* untuk melakukan pengujian aplikasi dengan melakukan *black box* testing agar mengetahui *input* dan *output* yang dihasilkan sesuai atau tidak.

Step V : Tahap terakhir adalah melakukan *Maintenance* yang berarti sistem yang sudah dirancang dapat dijalankan oleh *user* untuk mendeteksi plat kendaraan dan tetap dilakukan perbaikan apabila diperlukan.

3. HASIL DAN PEMBAHASAN

3.1 Perangkat Pendukung

Penelitian dilakukan dengan menggunakan perangkat komputer dengan spesifikasi yang tercantum pada Tabel 1 dan perangkat lunak yang digunakan untuk penelitian pada Tabel 2.

Tabel 1. Spesifikasi Laptop

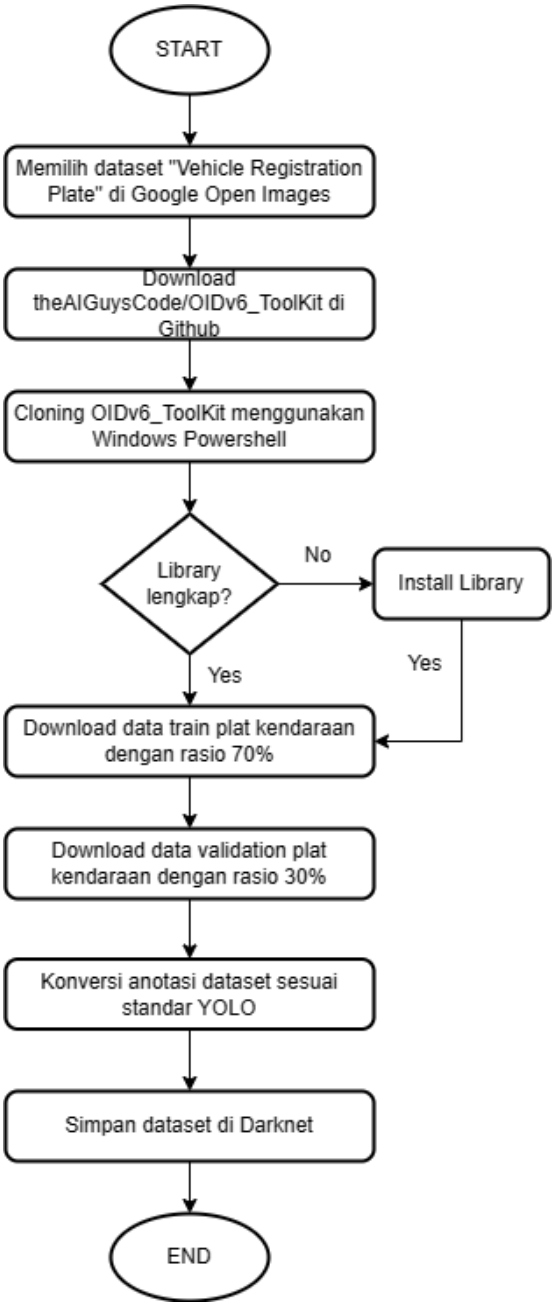
Aspek	Spesifikasi
Processor	APU AMD A12
RAM	4 GB
Storage	1 TB
Graphics Card	Radeon R7
Operating System	Windows 10 64-bit

Tabel 2. Spesifikasi Perangkat Lunak

Aspek	Spesifikasi
Bahasa Pemorgraman	Python, Java
Software Training Data	Google Collaboratory
Software Testing Data	Google Collaboratory
Command Prompt	Windows Powershell
Command Prompt	Anaconda Powershell Prompt
Software Pengkodean Aplikasi	Android Studio

3.2 Alur Pengerjaan Dataset

Pada penelitian ini terdapat dataset plat kendaraan yang memiliki data *train* dan data *validation*. Adapun proses pengambilan datasetnya dijelaskan pada Gambar 6 berikut:



Gambar 6. Alur Persiapan Dataset

Dataset yang sudah diunduh disimpan pada *Google Drive* yang terhubung dengan *Google Collaboratory*. Berikut merupakan jumlah data *train* dan *validation* yang di simpan pada *Google Drive*.

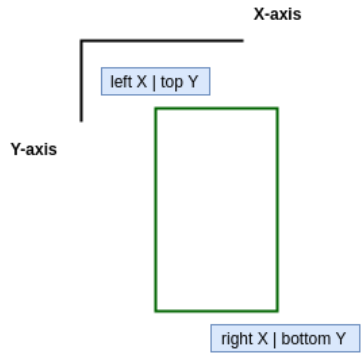
Tabel 3. Dataset

Label	Train	Validation
License Plate	1500	300

Gambar plat kendaraan yang digunakan oleh peneliti bersumber dari dataset plat kendaraan yang tersedia di *Google Open Images* karena sudah disediakan label serta anotasi pada bagian plat kendaraan sehingga lebih memudahkan proses pendeteksian bagian gambar yang merupakan plat kendaraan[16]. Kemudian dataset berupa plat kendaraan diunduh melalui *Windows Command Prompt* yang dibagi menjadi data *training* dan *validation* dimana setiap bagiannya akan dibagi menjadi 80% dan 20% . Hasil unduhan dataset merupakan plat kendaraan luar negeri. Kemudian dataset plat kendaraan yang sudah diunduh digunakan untuk proses implementasi membangun model pendeteksi plat kendaraan dengan YOLOv4.

Dataset yang sudah diunduh tidak hanya berupa gambar namun juga termasuk label gambar sesuai dengan jumlah gambar yang tersedia dengan format .txt yang akan digunakan untuk mengatur besar kecilnya objek dan rotasi yang dideteksinya[17]. Sebelumnya label hanya menunjukan nama class, Xmin, Xmax, Ymin, Ymax (koordinat kotak) tercantum pada Gambar 7 yang merupakan gambar dinormalisasi. Xmin ada di [0,1], dimana 0 merupakan piksel paling kiri, dan 1 merupakan piksel paling kanan pada gambar dan koordinat Y bergerak dari piksel atas (0) ke piksel bawah (1). Namun untuk mendapatkan titik koordinat yang lebih cakap dilakukan denormalisasi dengan menempatkan titik koordinat berpasangan di satu tempat sesuai dengan titiknya atau disebut dengan anotasi gambar. Sehingga menghasilkan setiap anotasi file .txt menjadi :

name_of_the_class left top right bottom



Gambar 7. Koordinat Kotak

Sebelum dilakukan anotasi label seluruh dataset masih di deskripsikan sebagai nama *class* nya dan menunjukan titik koordinat left X, right X, left Y, right Y. Berikut Gambar 8 merupakan tampilan sebelum dilakukan anotasi label.

```
Vehicle registration plate 226.100224 382.65984000000003 254.859264 394.04928
Vehicle registration plate 635.70944 580.257792 756.975616 638.397696
```

Gambar 8. Titik Koordinat Sebelum Anotasi

Kemudian setelah dilakukan anotasi, maka nama *class* akan dideskripsikan menjadi nilai 0 dan titik koordinat akan menjadi left X, top X, right Y, right Y. Berikut Gambar 9 setelah dilakukan anotasi label.

```
0 0.6930859375 0.5856246666666668 0.17125 0.11499999999999992
```

Gambar 9. Titik Koordinat Setelah Anotasi

3.3 Skenario Training Data

Pada proses ini sebelum melakukan proses *Transfer Learning* dibuat pengaturan menggunakan *Darknet* milik AlexeyAB yang merupakan *framework open-source* untuk jaringan syaraf sebagai arsitektur YOLOv4. Proses *Transfer Learning* di *Darknet* membutuhkan *file* konfigurasi dan *pre-trained weight* yang digunakan untuk membentuk jaringan untuk pelatihan dataset, dan *pre-trained weight* agar dapat mengenali objek baru. *Transfer Learning* bekerja dengan memanfaatkan model yang sudah dilatih untuk membuat model baru[18]. Pada *file* konfigurasi ditentukan *hyperparameter* seperti *batch size*, *subdivisions*, *max batch*, dan *classes* kemudian proses *training* dapat dimulai.

Hasil dari proses pelatihan data akan menghasilkan beberapa parameter yaitu nilai presisi, *recall*, F1-score, rata-rata IoU, dan mAP. Yang mana hasil dari *training* dataset ini adalah *file weights* yang akan digunakan untuk menguji program deteksi plat kendaraan.

Dataset yang digunakan menggunakan plat kendaraan luar negeri dan berasal dari internet dikarenakan situasi yang tidak kondusif untuk mengambil data secara langsung di publik sehingga menggunakan dataset yang diperoleh dari *Google Open Images* dengan nama kelas *"Vehicle Registration Plate"*.

Sementara itu dataset yang digunakan dibagi menjadi data *training* sebesar 80% dan data validation 20%. Data *training* ditentukan lebih besar daripada data *validation* untuk menghindari terjadinya *overfitting*[19],[20],[21]. Adapun konfigurasi *file* yang digunakan adalah sebagai berikut :

1. *Batch size* = 64
2. *Subdivisions* = 32
3. *Max batches* = 6000
4. *Steps* = 4800, 5400
5. *Classes* = 1
6. *Filters* = 18

Hasil dari *training* dataset didapatkan dengan parameter sebagai berikut :

1. Presisi = 0.85
2. *Recall* = 0.83
3. F1-*score* = 0.84
4. Rata-rata IoU = 65.46%
5. mAP = 86.82%

Adapun hasil training model yang dilakukan pada skenario dataset dapat ditujukan dari juga pengujian yang dilakukan dengan menggunakan data uji yang belum pernah dikenali oleh model pada Gambar 10 dan 11 setelah melakukan *training* model dengan YOLOv4 pada Gambar 11. Hasilnya pada Gambar 10 model dapat mendeteksi objek plat kendaraan pada banyak objek dalam satu gambar dengan beragam hasil akurasi pada gambar.

Gambar 10. Hasil Deteksi Pada Banyak Objek

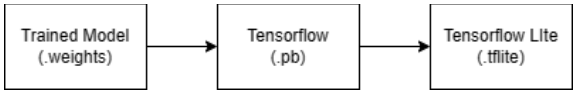
Gambar 11. Hasil Deteksi pada Plat Kendaraan (Google Open Images)

Pada pengujian prediksi plat kendaraan di Gambar 11 menunjukan hasil akurasi sebesar 0.77 atau 77% dengan waktu deteksi 0.05 detik. Hasil menunjukan bahwa model YOLOv4 yang menggunakan dataset *custom* berupa plat kendaraan dapat mendeteksi objek dengan sangat baik dan cepat.

Apabila dibandingkan dengan penelitian sejenis[4] yang menggunakan konfigurasi sama mendapatkan hasil mAP 72.68% saat menggunakan pre-trained weights yang dilatih sendiri.

3.4 Skenario Konversi Model

Untuk dapat menjalankan model terlatih YOLO pada aplikasi *android* membutuhkan konversi model menjadi format *Tensorflow Lite* agar dapat digunakan pada *android*. Adapun proses pengerjaannya dijelaskan pada Gambar 12 sebagai berikut :



Gambar 12. Alur Konversi Trained Model

Pada Gambar 12 dijelaskan sebagaimana pengerjaan konversi model terlatih YOLO adalah dengan mengubah menjadi *Tensorflow* terlebih dahulu yang mana hasilnya adalah *file Protobuf* (.pb) untuk selanjutnya diubah menjadi *Tensorflow*

Lite (.tflite) yang merupakan format model untuk standar penggunaannya pada aplikasi *android*[22]. Tahap konversi dilakukan dengan menggunakan *Anaconda Powershell Prompt*, dan untuk tahapannya menggunakan *repository* berjudul *Tensorflow-yolov4-tflite* dari pengguna *Github* bernama Hunglc007.

Proses *saved* model yang sudah selesai perlu dilakukan pengujian juga untuk mengetahui akurasi yang dihasilkan pada model *Tensorflow Lite*. Hasilnya sebagaimana yang ditujukan pada Gambar 13 dan Gambar 14 didapatkan hasil akurasi sebesar 100% dan 99% yang menunjukan model *TFLite* bekerja dengan sangat baik.

Gambar 13. Pengujian pada model Tensorflow Lite (Google Open Images)

Gambar 14. Pengujian ke dua pada model Tensorflow Lite (Kompasiana.com)

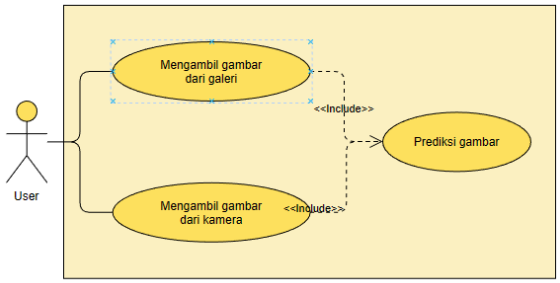
3.5 Perancangan Aplikasi Android

3.5.1 Metode Perancangan Aplikasi

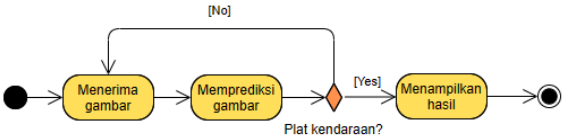
Dengan menggunakan metode *Waterfall* untuk rancang bangun sistem deteksi plat kendaraan, maka tahap-tahap dalam pelaksanaannya dapat dijelaskan dengan beberapa tahap mulai dari *Requirements, Design, Implementations, Verification, Maintenance.*

Requirements yang mana sistem deteksi plat kendaraan yang dibangun harus memiliki beberapa ketentuan yaitu dapat mengambil gambar secara langsung melalui kamera, mengakses galeri untuk memilih gambar yang akan dideteksi dan dapat mendeteksi plat kendaraan.

Design yang mana akan menjalankan apa yang sudah ditentukan pada tahap *Requirements* yaitu dengan memvisualisasikannya pada *Use Case Diagram* yang ditujukan pada Gambar 15 menjelaskan proses user menggunakan sistem aplikasi plat kendaraan dan *Activity Diagram* pada Gambar 16. menjelaskan mengenai alur proses sistem bekerja menghasilkan deteksi.

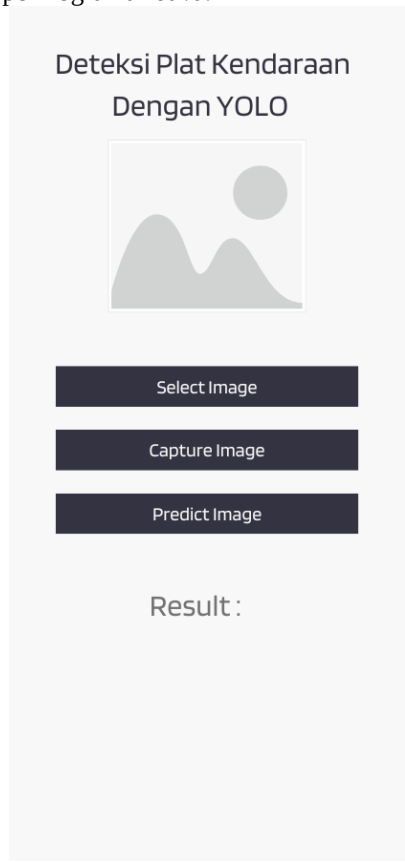


Gambar 15. Use Case Rancang Bangun Sistem



Gambar 16. Activity Diagram Rancang Bangun Sistem

Implementations yang mana pada tahap ini akan mengimplementasikan apa saja yang sudah ditentukan dan digambarkan pada tahap sebelumnya dengan menggambarkan *mockup* aplikasi untuk dibangun yaitu ditampilkan pada Gambar 17. Kemudian juga pada tahap implementasi sistem menggunakan perangkat lunak Android Studio dan bahasa pemrograman Java.



Gambar 17. Mockup Rancang Bangun Aplikasi

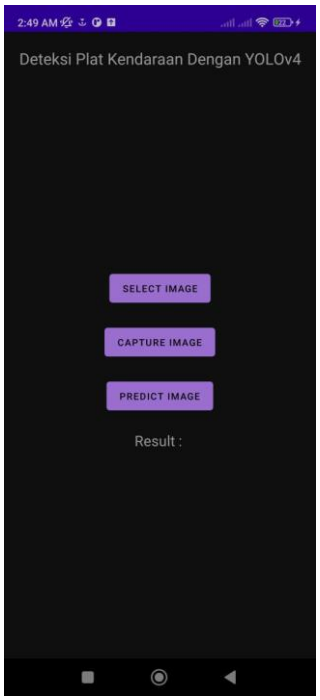
Verification yaitu dengan menguji aplikasi yang sudah dirancang sesuai dengan apa yang dibutuhkan pada tahap sebelumnya dan apakah *input* serta *output* yang dihasilkan sudah sesuai dengan proses *Design* dan sesuai dengan pengkodeannya. Adapun untuk melakukan pengujian ini menggunakan *blackbox testing* yang dijelaskan pada tabel 4.

Selanjutnya merupakan tahap terakhir yaitu *Maintenance* yaitu saat aplikasi sudah layak digunakan oleh *user* untuk mendeteksi plat kendaraan dan terus melakukan perbaikan apabila diperlukan pembaharuan.

3.5.2 Antarmuka Aplikasi Android

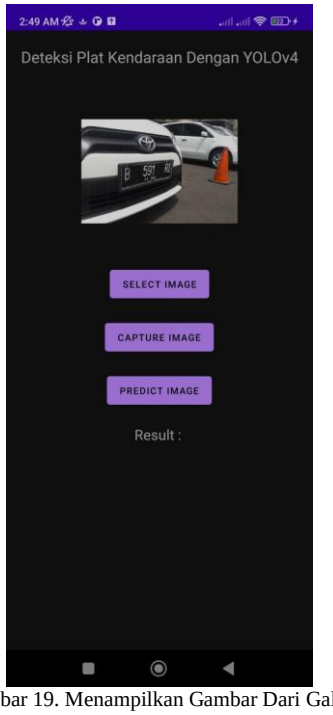
Pada pengerjaan aplikasi *android* menggunakan bahasa pemrograman *Java.* Aplikasi yang dibangun memiliki beberapa fitur diantaranya sebagai berikut :

- Halaman utama aplikasi, seperti yang disajikan pada Gambar 18 merupakan tampilan pertama ketika masuk ke aplikasi deteksi plat kendaraan. yang memiliki fitur *Select Image, Capture Image, Predict Image*.



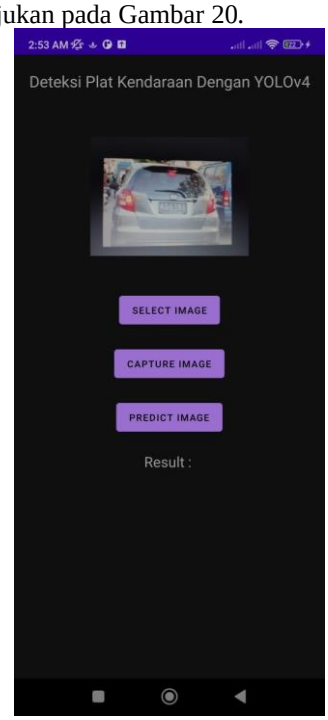
Gambar 18. Tampilan Awal Aplikasi

- *Select Image*, merupakan fitur yang digunakan untuk memilih gambar langsung dari perangkat pengguna untuk mendeteksi gambar pada aplikasi. Seperti yang ditujukan pada Gambar 19, aplikasi dapat mengakses galeri perangkat untuk mengambil gambar.



Gambar 19. Menampilkan Gambar Dari Galeri

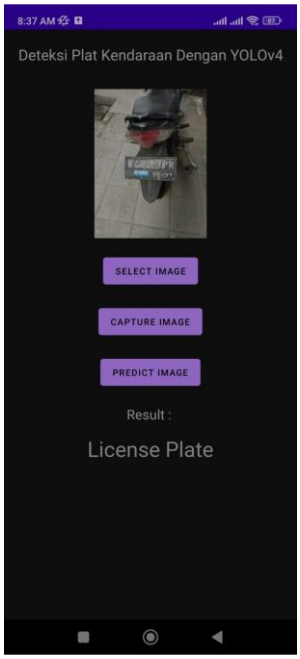
- *Capture Image*, merupakan fitur yang digunakan untuk mengambil gambar secara langsung di aplikasi dengan menggunakan kamera pada perangkat pengguna yang ditujukan pada Gambar 20.



Gambar 20. Mengambil Gambar dengan Kamera

- *Predict Image*, fitur yang digunakan untuk memprediksi gambar yang dihasilkan dari proses pemilihan gambar dari galeri maupun mengambil gambar secara langsung. Pada *predict image* digunakan untuk mengetahui apakah objek yang diambil merupakan gambar plat kendaraan.

Jika ya, maka akan dihasilkan sesuai dengan Gambar 21.



Gambar 21. Memprediksi Gambar

3.5.3 Pengujian Aplikasi

Selanjutnya setelah sistem deteksi plat kendaraan sudah dibangun, maka tahap selanjutnya merupakan pengujian aplikasi menggunakan *blackbox testing*. Adapun hasil pengujiannya sebagai berikut pada Tabel 4.

Tabel 4. Pengujian Aplikasi

Perintah	Aksi	Hasil	Keterangan
Memilih gambar dari galeri	Klik tombol "Select Image"	Membuka galeri untuk mengambil gambar	Berhasil
Mengambil gambar dari kamera	Klik tombol "Capture Image"	Membuka kamera untuk mengambil gambar	Berhasil
Memprediksi gambar yang diambil	Klik tombol "Predict Image"	Menunjukan hasil prediksi	Berhasil

Pada pengujian dengan *blackbox testing* didapatkan hasil bahwa semua fitur dapat dijalankan sesuai dengan perintah.

4. DISKUSI

Pada penelitian ini menggunakan parameter untuk menguji keberhasilan program yang dilatih tidak hanya dari rasio dataset yang digunakan, namun juga ada beberapa parameter lain yang turut mendukung keberhasilan pelatihan dataset yaitu dengan menentukan *batch size* sebesar 64, *subdivisions* sebesar 32, dan *max batch* sebesar 6000 dan tentunya *pre-trained weights* dari MS COCO yang digunakan untuk melakukan *transfer learning*.

Sehingga dari hasil pembahasan mengenai penelitian yang sudah dilakukan didapatkan hasil uji dataset berupa *file weights* yang dinilai beberapa parameternya menunjukkan keberhasilan model YOLO yang dibuat, yaitu presisi sebesar 0.85, *recall* sebesar 0.83, F1-*score* sebesar 0.84, serta rata-rata IoU 65,46%, dan diikuti dengan mAP sebesar 86.82% yang berarti hasil rata-rata akurasinya sangat baik karena diatas 80%. Kemudian mampu mendeteksi gambar dengan kelas *License Plate* sebesar 77% dengan waktu akurasi yang cukup singkat yaitu 0.05 detik.

Hasil model yang sudah dilatih juga dikembangkan menjadi format *Tensorflow Lite* yang mana dapat digunakan untuk membuat aplikasi deteksi plat kendaraan berbasis *android*, yang mana setelah dikonversi menjadi format *TFLite*, ternyata program mampu mendeteksi dengan mendapatkan akurasi terbaiknya sebanyak 100% seperti pada Gambar 13.

Apabila dibandingkan dengan penelitian terdahulu menjelaskan bahwa penelitian ini menggunakan metode YOLOv4 yang mana objek penelitiannya merupakan mendeteksi dengan melakukan perhitungan orang pada transportasi publik dan menghasilkan nilai mAP sebesar 72.68% namun dengan waktu deteksi yang tidak diketahui. Serta mampu mendeteksi dengan akurasi terbaiknya sebesar 95% pada kelas *human face*[4].

5. KESIMPULAN

Berdasarkan penelitian deteksi objek menggunakan algoritma YOLOv4 pada aplikasi *android* didapatkan kesimpulan dengan menggunakan parameter yang digunakan pada konfigurasi file sebelum dilakukannya penelitian maka pengujian yang dilakukan menghasilkan mAP sebesar 86.82% dengan kecepatan deteksi sebesar 0.05 detik sehingga dengan akurasinya yang sangat baik dan waktu deteksi yang cepat program dapat bekerja secara optimal untuk digunakan pada sistem deteksi plat kendaraan.

Selanjutnya dari sisi perancangan aplikasi *android* yang dibangun juga menunjukan hasil yang baik terbukti pada hasil program saat dijalankan dan pengujian yang dilakukan dengan *blackbox testing* bahwa aplikasi *android* dapat digunakan dengan baik dan layak.

Kemudian peneliti juga menyadari penelitian ini jauh dari kata sempurna oleh karena itu pada penelitian selanjutnya diharapkan dapat menambah fitur deteksi karakter pada plat kendaraan

DAFTAR PUSTAKA

[1] Badan Pusat Statistik, "Informasi Umum," 2022. https://www.bps.go.id/menu/1/informasi-umum.html#masterMenuTab5 (accessed Dec. 15, 2022).

[2] M. Michael, F. Tanoto, E. Wibowo, F. Lutan, and A. Dharma, "Pengenalan Plat Kendaraan Bermotor dengan Menggunakan Metode Template Matching dan Deep Belief Network", MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer, vol. 19, no. 1, pp. 27-36, Nov.2019.

[3] Badan Pusat Statistik, "Pengolahan Data," 2022. https://www.bps.go.id/menu/1/pengolahan-data.html#masterMenuTab5 (accessed Dec. 15, 2022).

[4] T. A. A. H. Kusuma, K. Usman, and S. Saidah, "PEOPLE COUNTING FOR PUBLIC TRANSPORTATIONS USING YOU ONLY LOOK ONCE METHOD", J. Tek. Inform. (JUTIF), vol. 2, no. 1, pp. 57-66, Feb. 2021.

[5] C. N. Liunanda, S. Rostianingsih, and A. N. Purbowo, "Implementasi Algoritma YOLO pada Aplikasi Pendeteksi Senjata Tajam di Android," *J. Infra*, vol. 8, no. 2, pp. 1–7, 2020.

[6] K. Khairunnas, E. M. Yuniarno, and A. Zaini, "Pembuatan Modul Deteksi Objek Manusia Menggunakan Metode YOLO untuk Mobile Robot," *J. Tek. ITS*, vol. 10, no. 1, pp. 50–55, 2021, doi: 10.12962/j23373539.v10i1.61622.

[7] F. Rofii, G. Priyandoko, M. I. Fanani, and A. Suraji, "Peningkatan Akurasi Penghitungan Jumlah Kendaraan dengan Membangkitkan Urutan Identitas Deteksi Berbasis Yolov4 Deep Neural Networks," *TEKNIK*, vol. 42, no. 2, pp. 169-177, Aug. 2021.

[8] E. Tirtana, K. Gunadi, and I. Sugiarto, "Penerapan Metode YOLO dan Tesseract-OCR untuk Pendataan Plat Nomor Kendaraan Bermotor Umum di Indonesia Menggunakan Raspberry Pi," *J. Infra*, vol. 9, no. 2, pp. 241–247, 2021, [Online]. Available:https://publication.petra.ac.id/index.php/teknik-informatika/article/view/11454

[9] M. D. Anggraini, K. Kusrini, and H. Al Fatta, "SOCIAL DISTANCING DETECTION FINDING OPTIMAL ANGLE WITH YOLO V3 DEEP LEARNING METHOD", *J. Tek. Inform. (JUTIF)*, vol. 3, no. 5, pp. 1449-1455, Oct. 2022

[10] J. Yu and W. Zhang, "Face MaskWearing Detection Algorithm Based on Improved YOLO-v4," *Sensors*, vol. 21, no. 9, pp. 1–21, 2022, doi: 10.1088/1742-6596/2258/1/012013.

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[12] N. Rochmawati, H. B. Hidayati, Y. Yamasari, H. P. A. Tjahyaningtijas, W. Yustanti, and A. Prihanto, "Analisa Learning Rate dan Batch Size pada Klasifikasi Covid Menggunakan Deep Learning dengan Optimizer Adam," *J. Inf. Eng. Educ. Technol.*, vol. 5, no. 2, pp. 44–48, 2021, doi: 10.26740/jieet.v5n2.p44-48.

[13] R. L. Pradana, A. Khumaidi, and R. Andiana, "Identifikasi Penyebab Cacat Pada Hasil Pengelasan Dengan Image Processing Menggunakan Metode YOLO," *J. Tek. Elektro dan Komput. Triac*, vol. 9, no. 3, pp. 1–5, 2022.

[14] G. Zhang, L. Ge, Y. Yang, Y. Liu, and K. Sun, "Fused Confidence for Scene Text Detection via Intersection-over-Union," in 2019 IEEE 19th International Conference on Communication Technology (ICCT), 2019, pp. 1540–1543, doi: 10.1109/ICCT46805.2019.8947307.

[15] D. H. Prayitna and A. Djajadi, "Perancangan Prototype Deteksi Kelengkapan," *J. Inov. Inform. Univ. Pradita*, vol. 7, no. 1, pp. 57–69, 2022.

[16] M. E. Laily, F. N. Fajri, and G. Q. O. Pratamasunu, "Deteksi Penggunaan Alat Pelindung Diri (APD) Untuk Keselamatan dan Kesehatan Kerja Menggunakan Metode Mask Region Convolutional Neural Network (Mask R-CNN)," *urnal Komput. Terap.*, vol. 8, no. 2, pp. 279–288, 2022.

[17] L. Rahma, H. Syaputra, A. H. Mirza, and S. D. Purnamasari, "Objek Deteksi Makanan Khas Palembang Menggunakan Algoritma YOLO (You Only Look Once)," *J. Nas. Ilmu Komput.*, vol. 2, no. 3, pp. 213–232, 2021, doi: 10.47747/jurnalnik.v2i3.534.

[18] V. Sowmya and R. Radha, "Comparative Analysis on Deep Learning Approaches for Heavy-Vehicle Detection based on Data Augmentation and Transfer-Learning techniques," *J. Sci. Res.*, vol. 13, no. 3, pp. 809–820, 2021, doi: 10.3329/jsr.v13i3.52332.

[19] A. Nurhopipah and U. Hasanah, "Dataset Splitting Techniques Comparison For Face Classification on CCTV Images," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 14, no. 4, pp. 341–352, 2020, doi: 10.22146/ijccs.58092.

[20] Y. A. Zebua *et al.*, "Prediksi Penetapan Tarif Penerbangan Menggunakan Auto-Ml Dengan Algoritma Random Forest," *J. Tek. Inf. dan Komput.*, vol. 5, no. 1, pp. 115–122, 2022, doi: 10.37600/tekinkom.v5i1.508.

[21] M. R. A. Yudianto, K. Kusrini, and H. Al Fatta, "Analisis Pengaruh Tingkat Akurasi Klasifikasi Citra Wayang dengan Algoritma Convolutional Neural Network," *J. Teknol. Inf.*, vol. 4, no. 2, pp. 182–191, 2020, doi: 10.36294/jurti.v4i2.1319.

[22] R. Roslidar, M. R. Syahputra, R. Muharar, and F. Arnia, "Adaptasi Model CNN Terlatih pada Aplikasi Bergerak untuk Klasifikasi Citra Termal Payudara," *J. Rekayasa Elektr.*, vol. 18, no. 3, pp. 185–192, 2022, doi: 10.17529/jre.v18i3.8754.