Pengembangan Model Deteksi Kendaraan Mobil di Area Parkir dengan Algoritma YOLOv5

Jose Agustian
Universitas Buana Perjuangan
Karawang, Indonesia
if18.joseagustian@mhs.ubpkarawang.ac.id

Tohirin Al Mudzakir
Universitas Buana Perjuangan
Karawang, Indonesia
tohirin@ubpkarawang.ac.id

Adi Rizky Pratama
Universitas Buana Perjuangan
Karawang, Indonesia
adi.rizky@ubpkarawang.ac.id

***Abstrak*—** Penggunaan kamera pengawas di area parkir dapat membantu memantau ketersediaan tempat parkir. Sulitnya mengetahui ketersediaan tempat parkir di fasilitas umum seperti pusat perbelanjaan dapat menyebabkan kerugian bagi pengemudi, seperti waktu terbuang dan konsumsi bahan bakar yang tidak efisien. Dengan mengetahui jumlah kendaraan yang telah berada di area parkir, informasi mengenai status ketersediaan tempat parkir dapat disediakan untuk mempermudah pengemudi. Penelitian ini bertujuan untuk mengembangkan model deteksi kendaraan mobil di area parkir serta mengevaluasi tingkat akurasi dan presisi model dalam mendeteksi kendaraan tersebut. Model deteksi dikembangkan dengan melatih model pra-latih YOLOv5s, yang kemudian diterapkan untuk mendeteksi kendaraan mobil pada area parkir. Hasil penelitian menunjukkan bahwa algoritma YOLOv5 efektif untuk mendeteksi kendaraan mobil di area parkir. Pada pengujian menggunakan 30 citra, model berhasil mendeteksi 914 kendaraan mobil dengan tingkat akurasi sebesar 90.59% dan tingkat presisi sebesar 94.85%.

***Kata kunci*—** Penglihatan Komputer, Deteksi Kendaraan Mobil, Algoritma YOLOv5

I. PENDAHULUAN

Area parkir adalah ruang yang digunakan untuk memarkirkan kendaraan, biasanya ditandai dengan kotak berwarna putih atau kuning. Umumnya terdapat di fasilitas publik seperti pusat perbelanjaan, dengan beberapa jenis area parkir seperti satu lantai, bertingkat, atau bawah tanah [1]. Penerapan sistem keamanan berbasis kamera pengawas pada area parkir dapat meningkatkan keamanan dan efisiensi pengelolaan parkir. Teknologi penglihatan komputer memungkinkan sistem cerdas untuk mendeteksi kendaraan dan memberi informasi ketersediaan tempat parkir, hal tersebut sangat penting karena pengemudi sering kesulitan mencari tempat parkir di area parkir yang luas atau tertutup. Survei Uber menunjukkan bahwa 74% responden di Jakarta mengkonfirmasi kesulitan ini sebagai penyebab keterlambatan, dengan rata-rata waktu pencarian area parkir mencapai 21 menit [2].

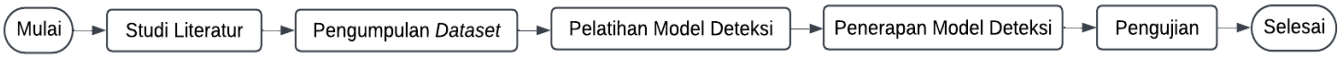
Telah dilakukan penelitian mengenai penglihatan komputer menggunakan metode Otsu dan algoritma *K-Nearest Neighbors* (KNN) untuk mengklasifikasi daging sapi berdasarkan warna, penelitian menunjukkan akurasi 74% untuk daging (92% sampel terdeteksi) dan 41% untuk lemak (86% sampel terdeteksi) dalam 50 percobaan [3]. Penelitian lain menggunakan ekstraksi fitur warna dan *euclidean distance* untuk klasifikasi kadar hidrasi tubuh berdasarkan warna urine dengan akurasi 75% dari 20 percobaan [4]. Selanjutnya, penelitian menggunakan algoritma YOLOv5 untuk mendeteksi area persawahan yang terdampak hama tikus, dengan akurasi 88% untuk area terdampak dan 70% untuk area tidak terdampak [5]. Terakhir, penelitian menggunakan *background substraction* dan *morphological operation* untuk memisahkan objek bergerak dari latar belakang dalam video pengawasan, memungkinkan deteksi gerak yang lebih akurat untuk menganalisis aktivitas normal atau mencurigakan melalui *bounding box* yang dihasilkan berdasarkan kecepatan dan rasio aspek tubuh yang telah ditentukan [6].

Kemudian, telah dilakukan penelitian terkait pencarian area parkir menggunakan algoritma *Modified* YOLO (M-YOLO) yang menunjukkan akurasi 100% dalam mendeteksi kendaraan di area parkir pada pengujian dengan 13 citra video [7]. Selanjutnya, penelitian menggunakan algoritma *Convolutional Neural Network* (CNN) dengan arsitektur VGG16 untuk mendeteksi hunian parkir, dilatih dengan *learning rate* 0.00001 dan 10 *epoch*. Hasil pengujian menunjukkan akurasi 0.9901 dan *loss* 0.0461 [8]. Terakhir, penelitian dengan menggunakan algoritma *Canny Edge* untuk mendeteksi lahan parkir dengan melakukan *masking* terhadap garis pada *region of interest*. Jumlah kendaraan dihitung dengan mengurangi kapasitas parkir yang tersedia, hasil pengujian menunjukkan akurasi tinggi dengan parameter tinggi kamera 40 cm, *threshold* 510, dan intensitas cahaya 100–115 *lux* [9].

Berdasarkan uraian di atas, diperlukan sebuah model yang dapat mendeteksi kendaraan mobil pada area parkir dengan akurat. Penelitian sebelumnya yang menggunakan algoritma M-YOLO [6] menunjukkan akurasi 100% dalam mendeteksi kendaraan, namun pengujian hanya dilakukan pada 13 citra dengan area parkir yang terbatas. Oleh karena itu, penelitian ini mengembangkan model deteksi kendaraan mobil pada area parkir menggunakan algoritma YOLOv5 dan melakukan pengujian pada 30 citra dengan area parkir yang lebih luas dan jumlah kendaraan yang lebih banyak. Hasil penelitian ini menunjukkan bahwa model yang dikembangkan dengan algoritma YOLOv5 mampu mendeteksi kendaraan mobil pada area parkir dengan tingkat akurasi sebesar 90.59% dan tingkat presisi sebesar 94.85%.

II. METODE PENELITIAN

Prosedur dari penelitian ini terdiri dari lima tahap yaitu studi literatur, pengumpulan *dataset*, pelatihan model deteksi kendaraan dengan YOLOv5, penerapan model deteksi pada citra area parkir, dan pengujian akurasi serta presisi menggunakan *tabel confusion matrix*. Gambar 1 akan menunjukkan bagaimana alur penelitian ini berjalan.



Gambar 1 Diagram Alir Penelitian

A. Studi Literatur

Studi literatur dilakukan untuk mencari informasi yang berkaitan dengan penelitian yang dilakukan. Studi literatur dilakukan dengan mencari bacaan sesuai dengan informasi yang ingin didapatkan, selain itu studi literatur juga dilakukan dengan mencari artikel jurnal penelitian terdahulu yang relevan dengan penelitian yang dilakukan.

B. Pengumpulan *Dataset*

Setelah studi literatur dilakukan, tahapan selanjutnya pengumpulan data citra area parkir dengan kendaraan mobil di dalamnya. Sebanyak 230 citra beresolusi 640 x 640 piksel diunduh dari situs Kaggle dan CNRPark. Gambar 2 akan menunjukkan contoh citra area parkir yang didapatkan dari kedua situs tersebut.

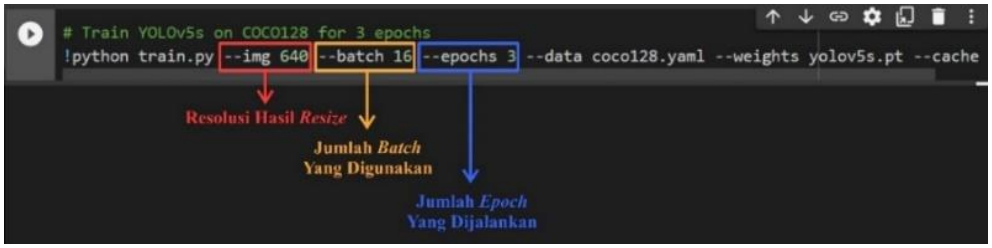
Gambar 2 Contoh Citra Area Parkir

Data citra tersebut kemudian dilabeli menggunakan layanan makesense.ai, hasil dari proses pelabelan tersebut menghasilkan berkas dengan format .txt untuk setiap citra. Berkas tersebut berisikan informasi nomor kelas objek, koordinat (pusat x, pusat y) serta ukuran (lebar, tinggi) setiap *bounding box* yang ada pada tiap citra tersebut.

C. Pelatihan Model Deteksi

Pada tahapan ini, hasil dari pengumpulan *dataset* digunakan untuk melakukan pelatihan model deteksi kendaraan mobil dengan menggunakan algoritma YOLOv5. Proses pelatihan dilakukan melalui *notebook* pada Google Colab guna mempercepat proses pelatihan dengan memanfaatkan *Graphics Processing Unit* (GPU) yang tersedia. Tahapan pertama adalah mengklonakan repositori YOLOv5 dari layanan GitHub pada Google Colab, repositori tersebut berisikan *notebook* yang dapat digunakan untuk melakukan proses pelatihan.

Tahapan selanjutnya adalah mengunggah folder *dataset* yang sudah disiapkan ke dalam Google Colab yang terdapat repositori YOLOv5 di dalamnya, folder ini mencakup berkas citra untuk data latih dan validasi, serta berkas anotasi atau label citra dalam format .txt untuk kedua jenis data tersebut. Setelah itu, dilakukan penyesuaian pada berkas konfigurasi pelatihan untuk mengatur lokasi *dataset*, jumlah kelas objek yang akan dilatih dan nama kelas objek yang akan dilatih. Langkah terakhir adalah melakukan pelatihan model deteksi dengan menjalankan *cell* untuk mengeksekusi skrip *train.py* yang sudah disediakan pada repositori tersebut. Gambar 3 akan menunjukkan *cell* untuk melakukan proses pelatihan.



Gambar 3 *Cell* Untuk Melakukan Pelatihan

Pada *cell* tersebut terdapat beberapa konfigurasi pelatihan yang dapat disesuaikan, seperti resolusi citra setelah proses *resize* pada gambar yang akan dilatih, jumlah *batch* yang digunakan, dan jumlah *epoch* yang akan dijalankan. Hasil dari proses pelatihan

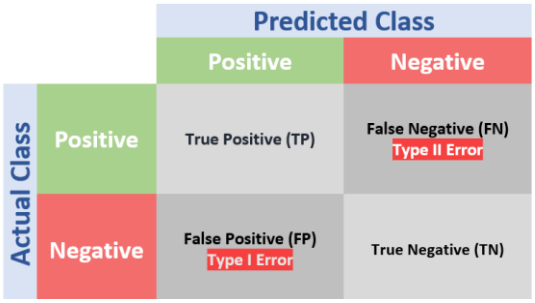
tersebut adalah dua *file* bobot yaitu *best.pt* yang mencatat bobot terbaik selama pelatihan yang nantinya akan digunakan untuk melakukan pendeteksian kendaraan mobil, serta *last.pt* yang berisikan bobot dari *epoch* terakhir yang dijalankan.

D. Penerapan Model Deteksi

Model yang dihasilkan dari proses pelatihan akan digunakan untuk melakukan pendeteksian, proses pendeteksian dapat dilakukan dengan mengeksekusi skrip *detect.py* melalui *cell* yang sudah disediakan pada *notebook* sebelumnya. Proses pendeteksian ini akan menghasilkan citra pengujian yang sudah diberi kotak pembatas (*bounding box*) pada setiap mobil yang terdeteksi. Citra hasil deteksi tersebut kemudian akan digunakan sebagai data pengujian.

E. Pengujian

Proses pengujian dilakukan dengan menggunakan tabel *confusion matrix* untuk mengetahui tingkat akurasi dan tingkat presisi model yang sudah dilatih, dari evaluasi ini dapat diketahui performa model deteksi dalam mendeteksi kendaraan mobil pada area parkir. Gambar 4 akan menunjukkan tabel *confusion matrix*.



Gambar 4 Tabel *Confusion Matrix*

III. HASIL DAN PEMBAHASAN

A. Studi Literatur

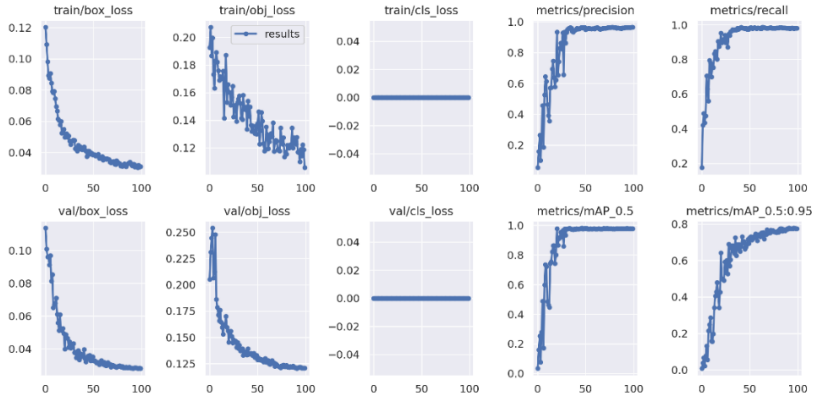
Berdasarkan dari studi literatur yang telah dilakukan, diketahui bahwa kecerdasan buatan dalam bidang penglihatan komputer memiliki kemampuan untuk melakukan deteksi objek menggunakan berbagai algoritma yang tersedia. Salah satu algoritma yang dapat digunakan adalah YOLOv5. Algoritma YOLOv5 ini dapat diterapkan untuk melatih model deteksi kendaraan, seperti mendeteksi mobil yang ada di area parkir.

B. Pengumpulan *Dataset*

Dataset citra kendaraan mobil pada area parkir diunduh dari Kaggle dan CNRPark, beresolusi 640 x 640 piksel dengan sudut pengambilan gambar yang bervariasi. Terdapat 230 citra yang dikelompokkan dalam folder *dataset*, yang terdiri dari folder *train_data* dan *data_test*. Folder *train_data* terbagi lagi menjadi folder *train* dan *val*, masing-masing berisi folder *images* (citra) dan *labels* (berkas teks pelabelan). Folder *train* berisi 160 citra untuk pelatihan, sementara folder *val* berisi 40 citra untuk validasi. Tahapan selanjutnya adalah pelabelan citra untuk melatih model. Proses pelabelan dilakukan menggunakan layanan makesense.ai, yang mempermudah proses ini. Langkah-langkah pelabelan mencakup mengunggah citra, membuat kelas objek, melabeli objek, dan mengekspor hasilnya dalam format berkas .zip. Hasil pelabelan berupa berkas teks yang mencakup kelas objek, koordinat, dan ukuran kotak pembatas untuk setiap objek dalam citra, dengan nilai skala 0–1, berkas teks hasil pelabelan akan disimpan pada folder *labels* yang sudah disebutkan sebelumnya. Selain itu, pada folder *data_test* terdapat 30 citra area parkir yang akan digunakan untuk pengujian model deteksi. Citra-citra pada folder *data_test* tidak digunakan dalam proses pelatihan atau validasi, sehingga hanya berfungsi sebagai data uji untuk mengevaluasi performa model setelah proses pelatihan selesai. Setelah seluruh *dataset* terkumpul dan direktori foldernya disesuaikan, folder *dataset* tersebut kemudian diunggah ke Google Colab.

C. Pelatihan Model Deteksi

Pelatihan model deteksi akan dilakukan menggunakan Google Colab, yang menyediakan GPU untuk mempercepat proses pelatihan dibandingkan dengan CPU. Langkah pertama adalah mengklonakan repositori YOLOv5, mengunggah *dataset* yang telah disiapkan, lalu membuat berkas konfigurasi dengan format .yaml yang berisi lokasi data latih dan validasi, jumlah objek yang akan dilatih, serta nama kelas objek. Setelah semua berkas siap, pelatihan model deteksi kendaraan mobil dimulai dengan menggunakan model pra-latih YOLOv5s, dikarenakan *dataset* yang terbatas. Pelatihan dilakukan dengan *batch size* 10, 16 iterasi per *epoch*, dan 100 *epoch* yang akan dijalankan. Setelah proses pelatihan model deteksi kendaraan mobil dilakukan, didapatkan hasil dari pelatihan seperti nilai *box loss*, *object loss*, *precision*, *recall*, dan nilai *mean Average Precision* (mAP) pada tiap *epoch*-nya. Gambar 5 akan menunjukkan hasil dari pelatihan model deteksi kendaraan mobil secara lengkap yang dituangkan ke dalam bentuk grafik.



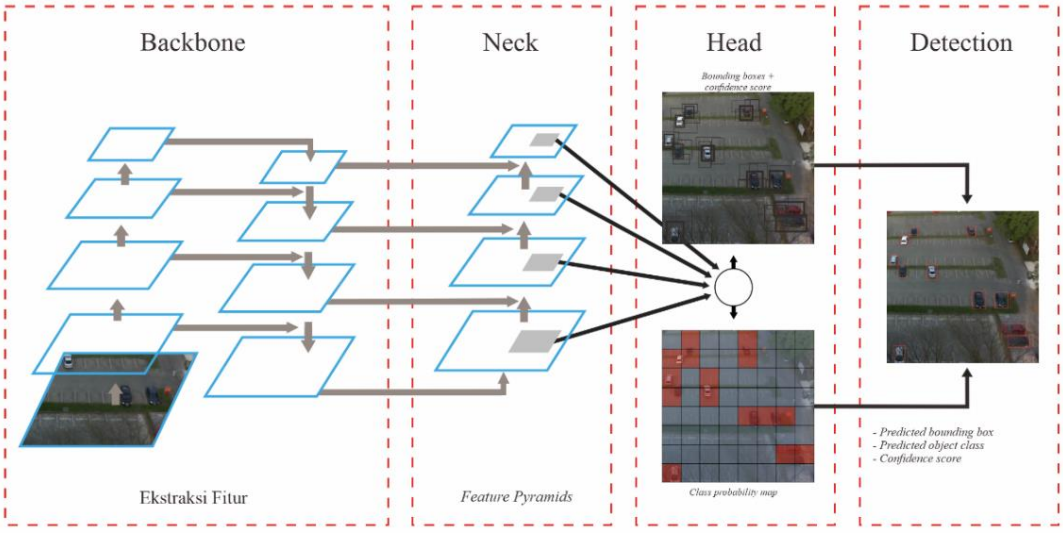
Gambar 5 Grafik Hasil Pelatihan

Berdasarkan grafik tersebut, hasil pelatihan menunjukkan bahwa model mencapai nilai presisi (*precision*) sebesar 0.963, nilai tersebut menunjukkan bahwa ketika model memprediksi suatu objek sebagai mobil, sekitar 96.3% dari prediksi tersebut adalah benar. Artinya, model sangat jarang salah mengidentifikasi objek lain sebagai mobil, yang menunjukkan kemampuannya dalam menghindari kesalahan klasifikasi *false positive*. Selain itu, nilai *recall* sebesar 0.979 menunjukkan bahwa model berhasil mengidentifikasi sekitar 97.9% dari semua kendaraan mobil yang terdapat pada citra. Dengan kata lain, model sangat efektif dalam mendeteksi seluruh kendaraan mobil yang ada, sehingga dapat menghindari *false negative* dengan baik. Terakhir, nilai mAP@.5 sebesar 0.978 menunjukkan performa keseluruhan model yang sangat baik dalam mendeteksi mobil pada area parkir.

Hasil pelatihan juga menunjukkan bahwa nilai *box_loss* pada data latih dan validasi sudah cukup rendah, yaitu di angka 0.03. Ini berarti model sudah cukup akurat dalam menggambar *bounding box* di sekitar mobil pada citra dan berhasil mengidentifikasi posisi mobil dengan baik. Namun, nilai *object_loss* pada data latih dan validasi masih cukup tinggi, yaitu 0.1, dan tidak dapat diperkecil lagi. Nilai tersebut menunjukkan bahwa model masih kesulitan dalam mengklasifikasikan objek yang terdeteksi sebagai mobil. Meskipun model sudah dapat menemukan posisi mobil, namun model masih sering salah dalam mengidentifikasi objek tersebut. Tingginya nilai *object_loss* mungkin disebabkan karena adanya kesalahan saat proses pelabelan citra pada *dataset*.

D. Penerapan Model Deteksi

Untuk memulai pengujian, perlu dilakukan deteksi kendaraan mobil terlebih dahulu terhadap 30 citra pengujian yang sudah disiapkan sebelumnya. Penerapan model deteksi dilakukan dengan menggunakan model yang sudah dilatih pada tahap pelatihan sebelumnya. Dalam melakukan deteksi, YOLOv5 menggunakan *Cross Stage Partial Network* (CSPNet) untuk melakukan ekstraksi fitur dari citra yang diberikan, kemudian setelah itu citra yang sudah diekstraksi akan diproses pada *Path Aggregation Network* (PANet). PANet itu sendiri memproses citra menjadi beberapa skala dan resolusi yang berbeda dengan cara melakukan *upsampling* pada citra, hal ini dilakukan agar model dapat mendeteksi objek yang sama meskipun dengan skala yang berbeda. Kemudian setelah itu model akan membentuk kumpulan kotak pembatas yang memenuhi citra, kemudian dilakukan *Non-Maximal Suppression* (NMS) untuk menghilangkan kotak pembatas hasil dari prediksi yang memiliki nilai lebih rendah dari ambang batas yang ditentukan, model akan memilih kotak pembatas yang memiliki nilai *objectness score* terbesar dan menghilangkan semua kotak pembatas yang serupa namun dengan nilai yang lebih rendah. Gambar 6 akan menunjukkan tahapan YOLOv5 dalam mendeteksi objek.



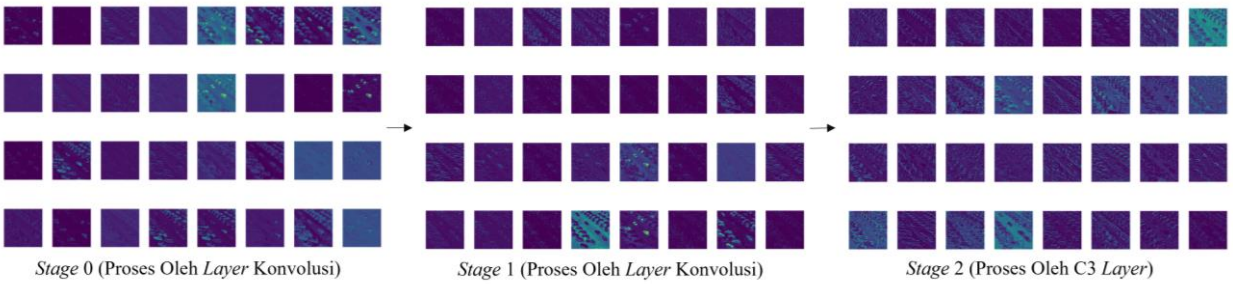
Gambar 6 Tahapan Deteksi Pada YOLOv5

Setiap tahapan YOLOv5 dalam mendeteksi objek mencakup ekstraksi fitur, penerapan *feature pyramids*, dan pendeteksian akan dijelaskan lebih lanjut sebagai berikut. Gambar 7 menunjukkan contoh citra yang digunakan untuk menjelaskan setiap tahapan deteksi.

Gambar 7 Contoh Citra Untuk Tahapan Deteksi

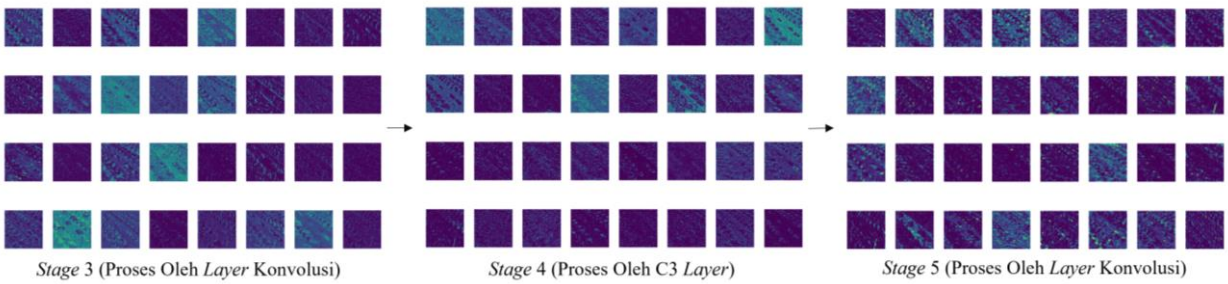
1) Ekstraksi Fitur

Pada tahapan ekstraksi fitur, model akan memproses citra masukkan (640 *x* 640 px, 3 *channel*) dengan serangkaian tahapan yaitu memproses citra dengan *layer* konvolusi (*Stage* 0) sehingga menghasilkan keluaran *feature map* beresolusi 320 *x* 320 *px* dengan 64 *channel*, kemudian dilanjutkan dengan pemrosesan dengan *layer* konvolusi (*Stage* 1) yang menghasilkan keluaran *feature map* beresolusi 160 *x* 160 *px* dengan 128 *channel*, lalu dilanjutkan dengan pemrosesan oleh CSP *Bottleneck* dengan 3 *layer* konvolusi yang selanjutnya akan disebut C3 *Layer* (*Stage* 2). Gambar 8 akan menunjukkan rangkaian proses tersebut.



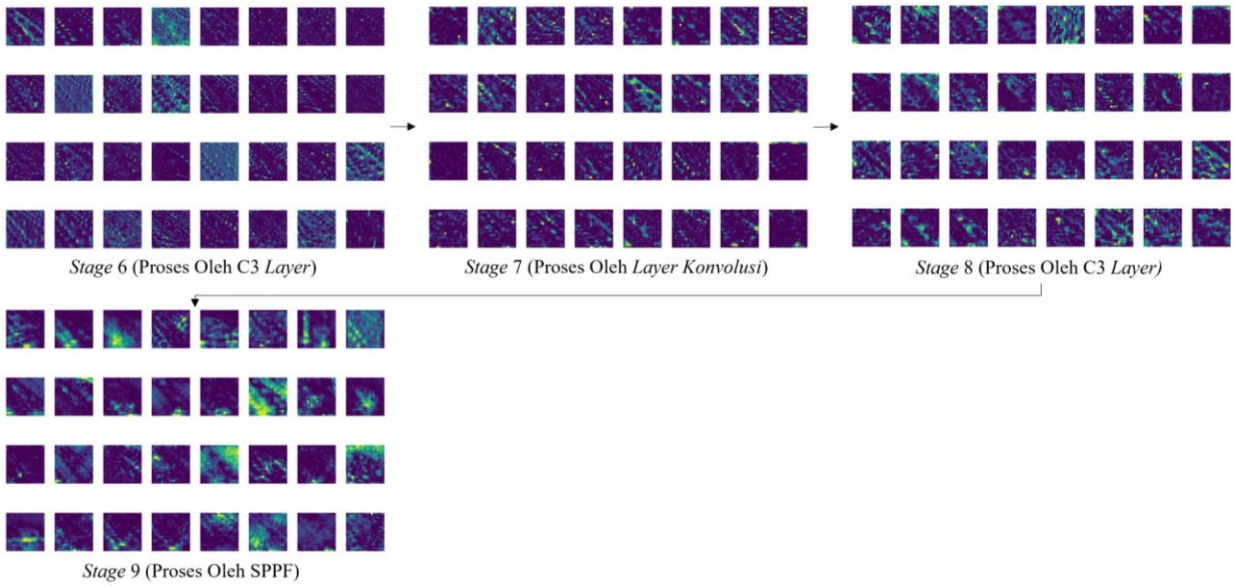
Gambar 8 Tahapan Pertama Ekstraksi Fitur

Feature map hasil dari *Stage* 2 dilanjutkan pemrosesan oleh *layer* konvolusi (*Stage* 3) untuk menghasilkan *feature map* beresolusi 80 *x* 80 *px* dengan 256 *channel*, kemudian *feature map* hasil dari *Stage* 3 dilanjutkan pemrosesan oleh C3 *layer* (*Stage* 4), kemudian *feature map* hasil dari *Stage* 4 dilanjutkan pemrosesan oleh *layer* konvolusi (*Stage* 5) yang akan menghasilkan *feature map* beresolusi 40 *x* 40 *px* dengan 512 *channel*. Gambar 9 akan menunjukkan rangkaian proses tersebut.



Gambar 9 Tahapan Kedua Ekstraksi Fitur

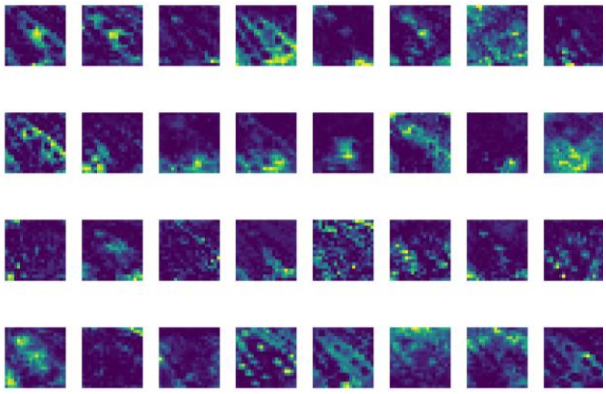
Feature map hasil dari *Stage* 5 diproses kembali oleh C3 *layer* (*Stage* 6), *feature map* hasil dari *Stage* 6 dilanjutkan dengan pemrosesan oleh *layer* konvolusi (*Stage* 7) sehingga akan menghasilkan *feature map* beresolusi 20 *x* 20 *px* dengan 1024 *channel*, kemudian *feature map* hasil dari *Stage* 7 akan diproses oleh C3 *layer* (*Stage* 8), *feature map* hasil dari *Stage* 8 akan diproses oleh *layer Spatial Pyramid Pooling-Fast* (SPPF) (*Stage* 9). Gambar 10 akan menunjukkan rangkaian proses tersebut.



Gambar 10 Tahapan Terakhir Ekstraksi Fitur

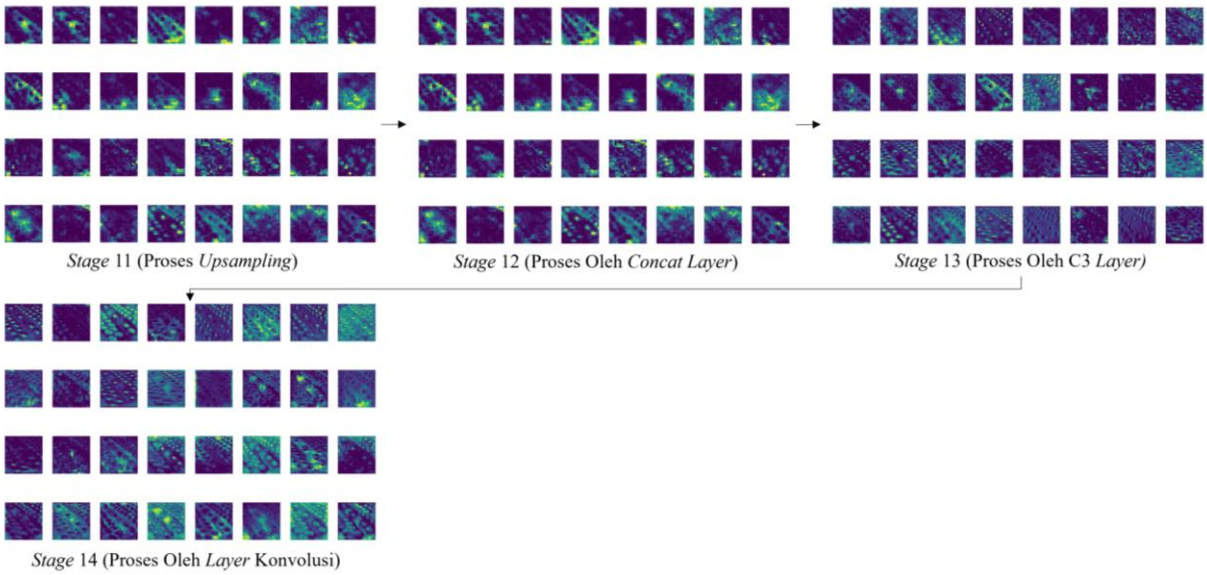
2) *Feature Pyramids* dengan PANet

Pada tahapan untuk menghasilkan *feature pyramids* ini, *Feature map* hasil dari *Stage* 9 pada tahapan sebelumnya akan diproses dengan *layer* konvolusi (*Stage* 10) untuk mengurangi jumlah *channel* pada *feature map* sebelumnya yang tadinya 1024 *channel* menjadi 512 *channel*. Gambar 11 akan menunjukkan tahapan pertama untuk menghasilkan *feature pyramids*.



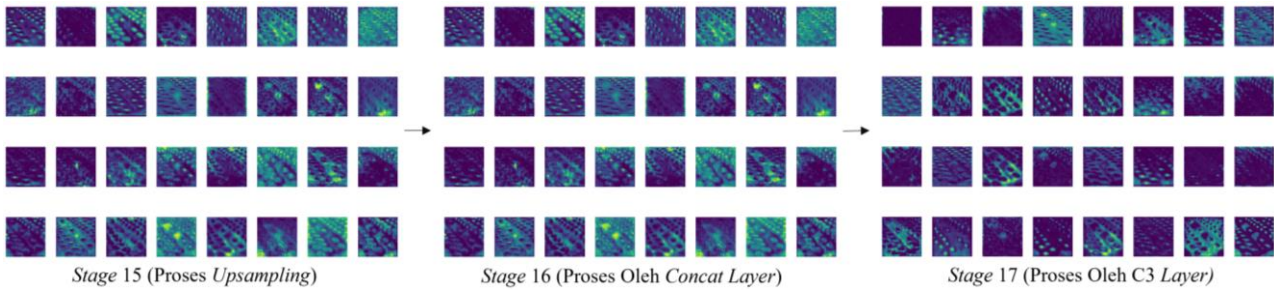
Gambar 11 *Stage* 10 (Hasil Proses Oleh *Layer* Konvolusi)

Feature map hasil dari *Stage* 10 dilanjutkan dengan proses *upsampling* (*Stage* 11) sehingga *feature map* yang tadinya memiliki resolusi 20 *x* 20 *px* menjadi 40 *x* 40 *px* dengan 512 *channel*, kemudian *feature map* hasil dari *Stage* 11 akan digabungkan dengan *feature map* hasil dari *Stage* 6 oleh *Concat layer* (*Stage* 12), hasil dari penggabungan tersebut akan diproses oleh C3 *layer* (*Stage* 13) dan diproses lanjutan oleh *layer* konvolusi (*Stage* 14) yang akan menghasilkan keluaran *feature map* beresolusi 40 *x* 40 *px* dengan 256 *channel*. Gambar 12 akan menunjukkan rangkaian proses tersebut.



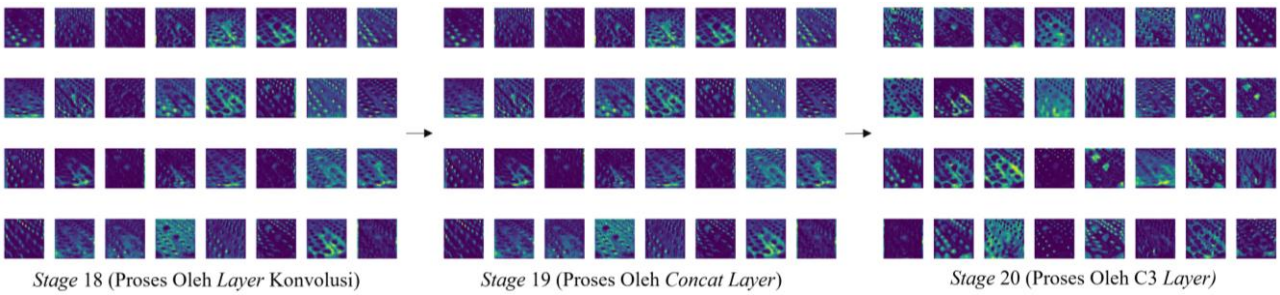
Gambar 12 Tahapan Kedua *Feature Pyramids*

Feature map yang dihasilkan pada *Stage* 14 akan di-*upsampling* sehingga akan menjadi *feature map* beresolusi 80 *x* 80 *px* dengan 256 *channel* (*Stage* 15), *feature map* tersebut akan digabungkan dengan *feature map* hasil dari *Stage* 4 oleh *Concat layer* (*Stage* 16), *feature map* yang dihasilkan pada *Stage* 16 akan diproses oleh C3 *layer* (*Stage* 17). Gambar 13 akan menunjukkan rangkaian proses tersebut.



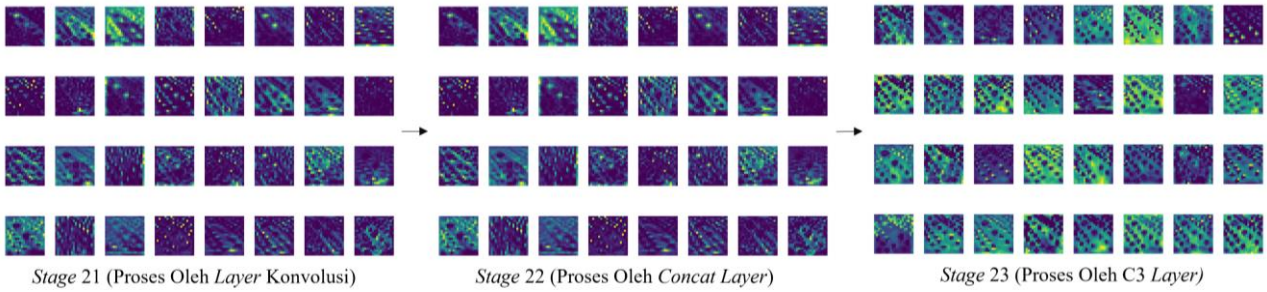
Gambar 13 Tahapan Ketiga *Feature Pyramids*

Feature map dari *Stage* 17 akan diproses oleh *layer* konvolusi (*Stage* 18) untuk menurunkan resolusi *feature map* yang tadinya 80 *x* 80 menjadi 40 *x* 40 *px*, *feature map* yang dihasilkan oleh *Stage* 18 akan digabungkan dengan *feature map* dari *Stage* 14 oleh *Concat layer* (*Stage* 19), *feature map* yang sudah digabungkan tersebut akan diproses oleh C3 *layer* (*Stage* 20) sehingga akan menghasilkan *feature map* beresolusi 40 *x* 40 *px* dengan 512 *channel*. Gambar 14 akan menunjukkan rangkaian proses tersebut.



Gambar 14 Tahapan Keempat *Feature Pyramids*

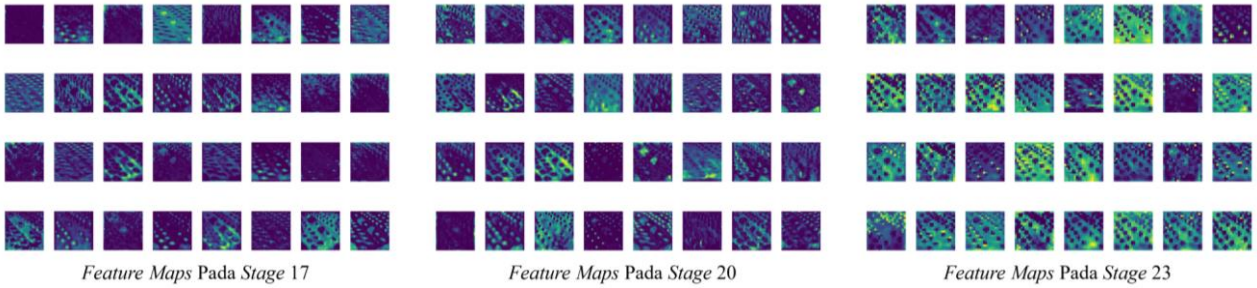
Feature map yang dihasilkan oleh *Stage* 20 akan diproses oleh *layer* konvolusi (*Stage* 21) untuk menurunkan resolusi *feature map* yang tadinya 40 *x* 40 *px* menjadi 20 *x* 20 *px*, kemudian *feature map* hasil dari *Stage* 21 digabungkan dengan *feature map* hasil dari *Stage* 10 oleh *Concat layer* (*Stage* 22). *Feature map* gabungan hasil dari *Stage* 22 akan diproses oleh C3 *layer* (*Stage* 23) sehingga akan menghasilkan *feature map* beresolusi 20 *x* 20 *px* dengan 1024 *channel*. Gambar 15 akan menunjukkan rangkaian proses tersebut.



Gambar 15 Tahapan Terakhir *Feature Pyramids*

3) Deteksi Kendaraan Mobil

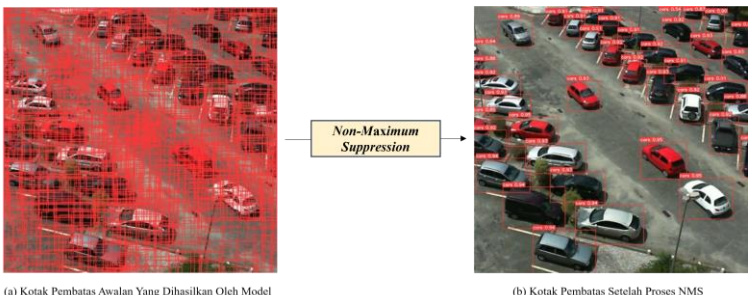
Pada tahapan deteksi, model akan menggunakan *feature map* yang dihasilkan oleh *feature pyramids* yaitu pada *Stage* 17 (80 *x* 80 *px*, 256 *channel*) yang selanjutnya akan disebut P3, *Stage* 20 (40 *x* 40 *px*, 512 *channel*) yang selanjutnya akan disebut P4, dan *Stage* 23 (20 *x* 20 *px*, 1024 *channel*) yang selanjutnya akan disebut P5. *Feature map* pada *Stage 17* digunakan untuk mendeteksi objek berskala kecil, *feature map* pada *Stage* 20 digunakan untuk mendeteksi objek berskala sedang, dan *feature map* pada *Stage* 23 digunakan untuk mendeteksi objek berskala besar. Gambar 16 akan menunjukkan tiap *feature map* yang digunakan untuk tahapan deteksi.



Gambar 16 *Feature Maps* Untuk Tahapan Deteksi

Dalam mendeteksi kendaraan mobil, model akan memecah citra ke dalam *N grid* menyesuaikan dengan ukuran *feature map* tersebut. Pada *feature map* P3 dengan 8 *stride*, jika citra masukkan memiliki resolusi 640 *x* 640 *px*, maka model akan memecah citra ke dalam 80 *grid* (640 *px* / 8). Pada *feature map* P4 dengan 16 *stride*, maka model akan memecah citra ke dalam 40 *grid* (640 *px* / 16). Pada *feature map* P5 dengan 32 *stride*, maka model akan memecah citra ke dalam 20 *grid* (640 *px* / 32). Dari tiap *grid* tersebut ketika terdapat pusat dari sebuah objek yang ada pada tiap *grid*, model bertanggung jawab untuk menghasilkan keluaran berupa kotak pembatas dengan nilai xywh (koordinat dan ukuran kotak pembatas), *objectness score*, dan kelas objek tersebut.

Setelah menghasilkan kotak pembatas, model akan menjalankan NMS (*Non-Maximum Suppression*) untuk menghilangkan kotak pembatas yang memiliki nilai ambang batas IOU (*Intersection of Union*) lebih tinggi dari yang ditentukan. Tahapan NMS dimulai dengan memilih kotak pembatas yang memiliki *objectness score* tertinggi, kemudian kotak pembatas yang dipilih akan dibandingkan dengan kotak pembatas lain, jika nilai IOU dari kedua kotak pembatas tersebut memiliki nilai di atas 0.45 (nilai ambang batas IOU pada model) maka kotak pembatas dengan nilai *objectness score* yang lebih rendah akan dihilangkan. Tahapan NMS akan diulang hingga setiap kotak pembatas dengan nilai *objectness score* yang rendah pada tiap objek yang terdeteksi akan dihilangkan. Gambar 17 akan menunjukkan kotak pembatas awalan yang dihasilkan oleh model dan kotak pembatas setelah melalui tahapan NMS.



Gambar 17 Proses NMS Untuk Menghasilkan Kotak Pembatas

E. Pengujian

Pengujian dilakukan untuk mengetahui tingkat akurasi dan tingkat presisi dari model deteksi kendaraan mobil yang sudah dilatih. Pengujian dilakukan dengan menggunakan tabel *confusion matrix*, di mana hasil dari penerapan model deteksi adalah citra

pengujian yang telah diberi kotak pembatas pada setiap objek kendaraan yang terdeteksi. Hasil dari pendeteksian kendaraan mobil pada area parkir yang dihasilkan oleh model akan dilakukan pengujian seperti yang ditampilkan pada Tabel 1 di bawah ini.

Tabel 1 Pengujian Model Deteksi

Data Citra	Jumlah Kendaraan Mobil		Hasil Pengujian					
Nama Berkas	*Ground Truth*	Hasil Deteksi	TP	TN	FP	FN	Total CF	*A*
TEST_01	39	40	39	0	1	0	40	0.98
TEST_02	28	29	28	0	1	0	29	0.97
TEST_03	35	34	34	0	0	1	35	0.97
TEST_04	43	44	43	0	1	0	44	0.98
TEST_05	10	10	10	0	0	0	10	1
TEST_06	14	12	12	0	0	2	14	0.86
TEST_07	9	5	5	0	0	4	9	0.56
TEST_08	15	11	11	0	0	4	15	0.73
TEST_09	38	34	34	0	0	4	38	0.89
TEST_10	12	10	10	0	0	2	12	0.83
TEST_11	51	55	51	0	4	1	56	0.91
TEST_12	11	13	11	0	2	0	13	0.85
TEST_13	84	88	82	0	6	2	90	0.91
TEST_14	85	84	80	0	4	5	89	0.90
TEST_15	31	28	27	0	1	3	31	0.87
TEST_16	17	17	17	0	0	0	17	1
TEST_17	20	20	20	0	0	0	20	1
TEST_18	30	31	30	0	1	0	31	0.97
TEST_19	18	20	18	0	2	0	20	0.90
TEST_20	22	29	22	0	7	0	29	0.76
TEST_21	14	15	14	0	1	0	15	0.93
TEST_22	20	17	13	0	4	6	23	0.57
TEST_23	18	19	18	0	1	0	19	0.95
TEST_24	20	20	18	0	2	0	20	0.90
TEST_25	11	14	10	0	4	1	15	0.67
TEST_26	25	27	25	0	2	0	27	0.93
TEST_27	62	59	58	0	1	4	63	0.92
TEST_28	70	70	70	0	0	0	70	1
TEST_29	55	54	52	0	2	3	57	0.91
TEST_30	6	5	5	0	0	1	6	0.83
Total	913	914	867	0	47	43	957	-

Pada kolom jumlah kendaraan mobil dalam tabel pengujian, *ground truth* merujuk pada jumlah kendaraan mobil yang teramati pada citra pengujian, sedangkan hasil deteksi adalah jumlah objek yang terdeteksi oleh model sebagai kendaraan mobil. Pada kolom hasil pengujian terdapat beberapa variabel yaitu TP (jumlah objek yang terdeteksi sebagai kendaraan mobil dan memang benar kendaraan mobil), TN (jumlah objek yang tidak terdeteksi sebagai kendaraan mobil dan memang benar bukan kendaraan mobil), FP (jumlah objek yang terdeteksi sebagai kendaraan mobil namun ternyata bukan kendaraan mobil), FN (jumlah objek yang tidak terdeteksi sebagai kendaraan mobil namun ternyata objek tersebut adalah kendaraan mobil), Total CF (nilai keseluruhan dari tiap variabel dalam *confusion matrix* di tiap citra pengujian yang didapatkan dari (1)), dan *A* (nilai akurasi yang didapatkan dari (2)).

$$Total\ CF = TP + TN + FP + FN \quad (1)$$

$$A = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (2)$$

Setelah mendapatkan nilai akurasi pada tiap citra pengujian, langkah selanjutnya adalah menghitung nilai akurasi dan nilai presisi pada model deteksi. Untuk menghitung nilai akurasi model deteksi dapat dilakukan dengan formula yang sama untuk menghitung akurasi pada tiap citra seperti pada (2).

$$A = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (2)$$

$$= \frac{(867 + 0)}{(867 + 0 + 47 + 43)}$$
$$= \frac{867}{957}$$
$$= 0.9059\ x\ 100$$
$$= 90.59\%$$

Untuk menghitung nilai presisi model deteksi, dapat dilakukan dengan membagi total keseluruhan *true positive* dengan total dari *true positive* ditambah *false positive* seperti pada (3).

$$P = \frac{TP}{TP+FP} \quad (3)$$
$$= \frac{867}{867 + 47}$$
$$= \frac{867}{914}$$
$$= 0.9485\ x\ 100$$
$$= 94.85\%$$

Berdasarkan Tabel 1, sebanyak 914 kendaraan mobil terdeteksi, dengan 867 objek yang benar memang mobil dan terdeteksi sebagai mobil (*true positive*), 47 objek yang bukan mobil namun terdeteksi sebagai mobil (*false positive*), dan 43 objek yang tidak terdeteksi sebagai mobil padahal sebenarnya merupakan mobil (*false negative*). Hasil pengujian menunjukkan model memiliki nilai akurasi 90.59% dan nilai presisi sebesar 94.85%. Nilai tersebut bukanlah nilai yang mutlak karena dapat berbeda sesuai dengan beberapa faktor seperti posisi objek, sudut pengambilan kamera, intensitas cahaya, dan banyaknya jumlah objek pada citra.

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan pada penelitian yang telah dilakukan, dapat disimpulkan bahwa penerapan algoritma YOLOv5 untuk deteksi kendaraan mobil dapat dimulai dengan pengumpulan *dataset* citra kendaraan mobil pada area parkir, diikuti dengan proses pelabelan, dan diakhiri dengan pelatihan menggunakan model pra-latih YOLOv5s. Hasil pelatihan dengan 200 citra beresolusi 640 *x* 640 piksel, *batch size* 10, dan 100 *epoch* menghasilkan model dengan performa yang sangat baik, hal tersebut ditunjukkan oleh nilai *mean Average Precision* (mAP) sebesar 0.978. Pengujian pada 30 citra menunjukkan akurasi nilai akurasi sebesar 90.59% dan nilai presisi sebesar 94.85%, model berhasil mendeteksi 914 kendaraan dengan 47 kasus *false positive* serta 43 *false negative*. Meskipun model menunjukkan presisi yang sangat baik, sensitivitas model masih perlu ditingkatkan untuk mengurangi kesalahan deteksi. Secara keseluruhan, meskipun akurasi dan presisi model sudah cukup baik, perbaikan pada proses pelatihan dan pengelolaan variasi objek dan lingkungan diperlukan untuk meminimalkan kesalahan dan meningkatkan performa deteksi.

B. Saran

Berdasarkan hasil yang diperoleh pada penelitian ini, terdapat beberapa hal yang perlu diperbaiki agar pendeteksian objek dapat mencapai akurasi yang lebih baik. Pertama, terkait dengan tahapan pelatihan, disarankan untuk memulai dengan 300 *epoch*. Jika *overfitting* terjadi lebih awal, jumlah *epoch* dapat dikurangi, namun jika *overfitting* tidak terjadi setelah 300 *epoch*, maka jumlah *epoch* dapat ditingkatkan. Kedua, dalam hal pendeteksian objek, perbaikan *dataset* sangat diperlukan agar model dapat mendeteksi kendaraan dengan lebih akurat. Perbaikan tersebut mencakup peningkatan akurasi penempatan kotak pembatas atau label pada tahap pelabelan, di mana tidak boleh ada ruang antara objek dan kotak pembatas, serta penambahan citra tanpa objek (*background images*) sebanyak 0-10% dari total *dataset* untuk mengurangi nilai *false positive*.

PENGAKUAN

Naskah ilmiah ini adalah bagian dari penelitian Tugas Akhir milik Jose Agustian yang dibimbing oleh Tohirin Al Mudzakir dan Adi Rizky Pratama dengan judul Penerapan Algoritma YOLOv5 Untuk Deteksi Kendaraan Mobil Pada Area Parkir.

DAFTAR PUSTAKA

[1] H. Silderhuis, "Parking Facilities," Parking Network, 16 Agustus 2013. [Daring]. Tersedia: https://www.parking.net/about-parking/parking-facilities. [Diakses 6 Oktober 2022].

[2] Tim Brilio, "Berapa Uang & Waktu Terbuang Karena Susah Cari Parkir Di Jakarta?," brilio.net, 19 Februari 2019. [Daring]. Tersedia: https://www.brilio.net/serius/berapa-uang-waktu-terbuang-karena-susah-cari-parkir-di-jakarta-190218o.html. [Diakses 15 Januari 2022].

[3] A. R. Pratama, A. R. Juwita dan T. A. Mudzakir, "Klasifikasi Daging Sapi Berdasarkan Ciri Warna Dengan Metode Otsu dan K-Nearest Neighbor," *Techno Xplore,* vol. 6, no. 1, pp. 9-18, 2021.

[4] D. Wahiddin dan J. Indra, "Klasifikasi Kadar Hidrasi Tubuh Berdasarkan Warna Urine dengan Metode Ekstraksi Fitur Warna dan Euclidean Distance," *Techno Xplore,* vol. 5, no. 1, pp. 16-20, 2020.

[5] K. A. Baihaqi dan C. Zonyfar, "Deteksi Lahan Pertanian Yang Terdampak Hama Tikus Menggunakan YOLO v5," *Syntax: Jurnal Informatika,* vol. 11, no. 2, pp. 1-9, 2022.

[6] A. Fauzi, S. Madenda, Ernatusti, E. P. Wibowo dan A. F. N. Masruriyah, "The Importance of Bounding Box in Motion," dalam *2020 Fifth International Conference on Informatics and Computing (ICIC)*, Gorontalo, Indonesia, 2020.

[7] S. Jupiyandi, F. R. Saniputra, Y. Pratama, M. R. Dharmawan dan I. Cholissodin, "Pengembangan Deteksi Citra Mobil Untuk Mengetahui Jumlah Tempat Parkir Menggunakan CUDA dan Modified YOLO," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* vol. 6, no. 4, pp. 413-419, 2019.

[8] F. Assidhiqi, R. Rahmadi dan R. A. Rajagede, "Pengembangan Sistem Deteksi Hunian Parkir Menggunakan Metode Convolutional Neural Network," *Automata,* vol. 2, no. 1, pp. 224-231, 2021.

[9] F. P. Putra dan I. Susilawati, "Prototipe Sistem Deteksi Ketersediaan Lahan Parkir Menggunakan Metode Algoritma Canny Edge," *Journal of Information System and Artificial Intelligence (JISAI),* vol. 1, no. 2, pp. 94-99, 2021.