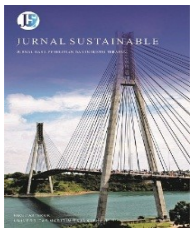


Jurnal Sustainable: Jurnal Hasil Penelitian dan Industri Terapan
Vol. 09, No. 01, hal. 8- 14, Mei 2020
Jurnal Sustainable: Jurnal Hasil Penelitian dan Industri Terapan
ISSN 2615-6334 (Online)
ISSN 2087-5347 (Print)

Deteksi Kendaraan Secara *Real Time* Menggunakan Metode YOLO Berbasis Android

Junita Sri Wisna H[1,*], Tekad Matulatan[2], Nurul Hayaty[3]
[1,2,3]Jurusan Informatika, Fakultas Teknik, Universitas Maritim Raja Ali Haji
Jl. Politeknik Senggarang, Tanjungpinang 29100
[*]Corresponding Author: junitasriwisna18@gmail.com

Abstract—The Activities on the highway that involve vehicles often have congestion problems due to the tightening of the quantity of vehicles on the highway. In addition, there are also problems of order and violations, the use of improper routes, such as vehicles entering the lane that are not intended for the vehicle. Therefore, researchers designed a vehicle detection application in real time based on Android using the YOLO method (You Only Look Once). The analysis carried out using 200 datasets, 4 classes, 10 batches, and 200 epochs. The training process was carried out up to 4000 steps, and the storage of checkpoints to the form of protobuf file was done at steps 800, 1000, 1200, 1400, 1600, 1800, 2000, 3000, and 4000. Bounding boxes successfully detected and classified objects correctly. This test is done using a xiaomi redmi 4X smartphone with a video resolution measuring 768x432 pixels.
Keywords—YOLO (You Only Look Once), Real Time, Bounding Box, Training, Testing.

Intisari—Aktivitas di jalan raya yang melibatkan kendaraan seringkali terdapat masalah kemacetan akibat memadatnya kuantitas kendaraan yang berada di jalan raya. Selain itu, ada pula masalah ketertiban dan pelanggarannya, penggunaan jalur yang tidak pada tempatnya, misalnya seperti kendaraan yang masuk ke jalur yang tidak diperuntukkan bagi kendaraan tersebut. Oleh sebab itu, peneliti merancang sebuah aplikasi deteksi kendaraan secara *real time* berbasis android menggunakan metode YOLO (*You Only Look Once*). Analisa yang dilakukan menggunakan jumlah dataset sebanyak 200, 4 kelas, 10 *batch*, dan 200 *epoch*. Proses pelatihan dilakukan hingga 4.000 *step*, dan penyimpanan *checkpoint* ke bentuk *protobuf file* dilakukan pada *step* 800, 1.000, 1.200, 1.400, 1.600, 1.800, 2.000, 3.000, dan 4.000. *Bounding box* berhasil mendeteksi dan mengklasifikasi objek secara tepat. Pengujian ini dilakukan menggunakan perangkat *smartphone* xiaomi redmi 4X dengan resolusi video berukuran 768x432 piksel.
Kata kunci—YOLO (*You Only Look Once*), *Real Time*, Kotak Pembatas, Pelatihan, Pengujian.

I. PENDAHULUAN

Menurut UU Republik Indonesia Nomor 22 Tahun 2009 Tentang Lalu Lintas dan Angkutan Jalan, kendaraan adalah suatu sarana angkut di jalan yang terdiri atas kendaraan bermotor dan kendaraan tidak bermotor. Kendaraan bermotor adalah setiap kendaraan yang digerakkan oleh peralatan mekanik berupa mesin selain kendaraan yang berjalan di atas rel. Sedangkan kendaraan tidak bermotor adalah setiap kendaraan yang digerakkan oleh tenaga manusia atau hewan.

Aktivitas yang seringkali terjadi di jalan raya yang melibatkan kendaraan-kendaraan, baik dalam kuantitas yang banyak maupun sedikit memiliki permasalahan salah satunya adalah

ISSN 2615-6334 (Online)
ISSN 2087-5347 (Print)

masalah kemacetan akibat memadatnya kuantitas kendaraan yang berada di jalan raya. Selain itu, ada pula masalah yang sering ditimbulkan ialah kurangnya ketertiban dan penggunaan jalur yang tidak pada tempatnya, misalnya seperti kendaraan yang masuk ke jalur yang tidak diperuntukkan bagi kendaraan tersebut. Oleh karena itu, pendeteksian kendaraan dapat digunakan untuk membantu memantau kondisi jalan raya dan memantau pelanggaran pada penggunaan jalur khusus.

Dalam penelitian ini, menggunakan sebuah metode dalam hal pendeteksian secara *real time*. Metode yang digunakan adalah YOLO (*You Only Look Once*). Metode YOLO adalah salah satu metode *state-of-the-art* untuk kasus pendeteksian objek dalam kondisi *real-time*. YOLO merupakan detektor dengan model terpadu (*unified*) yang memprediksi kotak pembatas dan probabilitas kelas secara langsung pada satu gambar penuh dalam satu kali evaluasi. Model dasar YOLO dapat memproses gambar pada 45 FPS (*frame per second*) pada kondisi *real-time* [1]. Namun, metode YOLO masih jauh dari sempurna untuk diterapkan pada *autonomous driving*, karena kesalahan-kesalahan yang dapat terjadi pada ukuran kotak pembatas menyebabkan terbatasnya penentuan jarak objek yang dideteksi [2].

Terdapat beberapa penelitian sebagai bahan pertimbangan dalam penelitian ini, berikut penelitian terdahulu yang terkait dalam penelitian ini yang telah dilakukan sebelumnya [3] menjelaskan implementasi metode YOLO untuk mendeteksi orang. Basis metode YOLO yang digunakan adalah YOLO dengan 7 lapisan konvolusional.

[4], mereka membangun sistem pengenalan tindakan manusia dengan menggunakan metode YOLO. Dataset yang digunakan adalah *Liris Human Activities dataset*, yang merupakan dataset berbentuk video. Model yang digunakan mengambil frame input setelah jangka waktu tertentu, dan memberikan label berdasarkan *frame* tunggal. [5], menjelaskan penerapan *Fast* YOLO pada deteksi objek yang tersemat di video secara *real-time* menggunakan dataset *Pascal VOC 2007*. Walaupun YOLOv2 dapat mencapai performa *real-time* pada GPU yang kuat, namun masih tetap menjadi tantangan untuk memanfaatkannya pada perangkat dengan daya komputasi dan memori yang terbatas. *Fast* YOLO diajukan dengan mengakselerasi YOLOv2, untuk mengurangi konsumsi daya pada perangkat.

[2] menjelaskan detektor objek dengan YOLO untuk *onboard driving*, menggunakan dataset *Udacity*, *KITTI Benchmark*, dan *Pascal VOC 2007*. Namun hasilnya menunjukkan bahwa YOLO memang merupakan salah satu sistem pendeteksian yang menjanjikan, tapi dikatakan masih jauh dari sempurna untuk diterapkan pada *autonomous driving*.

Perbedaan mendasar pada penelitian ini dari penelitian-penelitian tersebut adalah pada masalah yang akan diselesaikan. Pada penelitian ini dengan adanya Aplikasi Deteksi Kendaraan Secara *Real-Time* yang bertujuan untuk melakukan pendeteksian kendaraan di jalan raya akan menggunakan metode YOLO sebagai detektor dan untuk pengambilan data latih serta pengujiannya menggunakan android sedangkan untuk implementasinya menggunakan kamera pemantau lalu lintas di jalan raya.

II. BAHAN DAN METODE

A. YOLO (You Only Look Once)

YOLO (*You Only Look Once*) adalah sebuah pendekatan baru untuk sistem pendeteksian objek, yang ditargetkan untuk pemrosesan secara *real-time*. YOLO membingkai pendeteksian objek sebagai masalah regresi tunggal, dimana dari piksel gambar langsung ke kotak pembatas (*bounding box*) spasial yang terpisah dan probabilitas kelas yang terkait. YOLO melakukan pendeteksian dan pengenalan objek dengan sebuah jaringan syaraf tunggal (*single neural network*), yang memprediksi kotak-kotak pembatas dan probabilitas kelas secara langsung dalam satu evaluasi [1].

YOLO membagi gambar input menjadi *grid* $S \times S$. Jika pusat suatu objek jatuh ke dalam sel *grid*, maka sel *grid* bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel *grid* memprediksi kotak pembatas B dan nilai keyakinan (*box confidence scores*) untuk kotak

tersebut, serta probabilitas kelas kondisional *C*. Prediksi kotak pembatas *B* memiliki 5 komponen *x*, *y*, *w*, *h*, dan *box confidence score*. Koordinat (*x*, *y*) mewakili pusat kotak, relatif terhadap batas-batas kotak *grid*. Koordinat ini dinormalisasi untuk jatuh di antara 0 dan 1. Dimensi kotak (*w*, *h*) relatif terhadap ukuran gambar, dan juga dinormalisasikan. *Box confidence score* (nilai keyakinan) mencerminkan seberapa yakinnya model, bahwa kotak pembatas *B* berisi suatu objek dan seberapa akurat menurutnya bahwa kotak itu yang ia prediksi. Oleh karena itu, prediksi YOLO memiliki bentuk vektor *output* [*S*, *S*, *B*×5+*C*].

YOLO mendefinisikan nilai *confidence* untuk setiap kotak *B* sebagai nilai probabilitas kotak berisi objek yang dikalikan dengan *IoU* (*intersection over union*) antara kotak prediksi dan kebenaran dasar (*ground truth*), dimana kebenaran dasar didapat selama masa training. *IoU* adalah pengukuran evaluasi yang digunakan untuk mengukur seberapas akuratnya pendeteksi objek pada suatu dataset. Perhitungan *box confidence score* atau nilai *confidence* untuk sebuah kotak ditunjukkan pada Persamaan (1) dan Persamaan (2).

$$box\ confidence\ score = P_r(object) \cdot IoU_{pred}^{truth} \quad (1)$$

$$IoU_{pred}^{truth} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (2)$$

$\boldsymbol{P_r(object)}$: probabilitas kotak berisi objek, jika ada bernilai 1, sebaliknya bernilai 0.

$\boldsymbol{IoU_{pred}^{truth}}$: *IoU* Antara kotak prediksi dan kebenaran dasar.

Untuk mendapatkan prediksi final, faktor penentunya adalah *class confidence score* yang didapat, berdasarkan probabilitas kondisional kelas dan *box confidence score*. *Class confidence score* mengukur nilai kepercayaan pada klasifikasi dan lokalisasi objek. *Class confidence score* memberi nilai kepercayaan kelas spesifik untuk setiap kotak, yang mengkodekan kemungkinan kelas yang muncul di kotak dan seberapa sesuainya kotak yang diprediksi dengan objek. Persamaan pada *class confidence score* untuk setiap kotak prediksi ditunjukkan pada Persamaan (3).

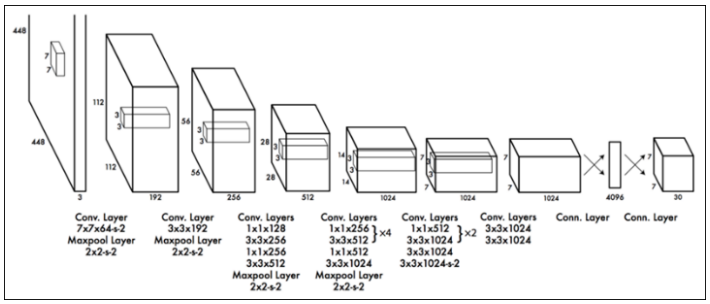
$$P_r(Class_i|object) \cdot box\ confidence\ score = P_r(Class_i) \cdot IoU_{pred}^{truth} \quad (3)$$

$\boldsymbol{P_r(Class_i|object)}$: probabilitas kondisional kelas *i*.

$\boldsymbol{P_r(Class_i)}$: probabilitas kelas *i*.

B. Arsitektur YOLO

Arsitektur YOLO terdiri atas 24 lapisan konvolusional (*convolutional layer*) dengan 4 lapisan *max pooling*, yang diikuti oleh 2 lapisan yang terhubung penuh (*fully connected layer*). Beberapa lapisan konvolusi menggunakan lapisan reduksi 1×1 sebagai alternatif untuk mengurangi kedalaman *feature maps* [1]. Arsitektur yang diperkenalkan oleh Joseph Redmon ditunjukkan seperti pada Gambar 1.



Gambar 1. Arsitektur YOLO

Arsitektur YOLO sesungguhnya cukup sederhana. Sistem akan menerima input citra dengan bentuk (448, 448, 3) yaitu citra berukuran 448 x 448 dengan 3 *channel*, yang kemudian akan melewati satu kali proses *convolutional network* hingga menghasilkan output dengan bentuk (7, 7, 30), dimana 7 x 7 merupakan ukuran grid sel (*S* = 7) dan 30 merupakan nilai dari jumlah kotak pembatas *B* yang dikali dengan penjumlahan antara jumlah kelas dan jumlah komponen dalam satu kotak *B* (*B*×5+*C*, *B* = 2, *C* = 20).

Setiap operasi konvolusi, selain memiliki parameter ukuran *kernel filter* dan jumlah *filter*, juga terdapat parameter lainnya yang mempengaruhi bentuk dari *output* hasil operasi konvolusi, yaitu *padding* dan *stride*. *Padding* adalah parameter yang menyatakan jumlah

ISSN 2615-6334 (Online)
ISSN 2087-5347 (Print)

penambahan *border* di seluruh tepian dari *input*, yang berfungsi menimimalisir kehilangan informasi pada tepian citra (*input*). Hal ini dikarenakan proses konvolusi itu sendiri, dimana biasanya bagian tepi dari citra akan terlewati oleh *kernel filter*, kecuali untuk *kernel* berukuran 1x1. Metode populer dan paling sederhana untuk mengatasi masalah ini adalah penggunaan *zero-padding*, yaitu menambahkan nilai 0 pada setiap tepian citra (*input*). Sedangkan *stride* adalah parameter yang menyatakan jumlah pergeseran (langkah) yang dilakukan oleh *kernel filter*. Parameter ini biasa digunakan untuk mereduksi ukuran *output*.

Ukuran *output* dari operasi konvolusi dapat dihitung dengan menggunakan persamaan (4).

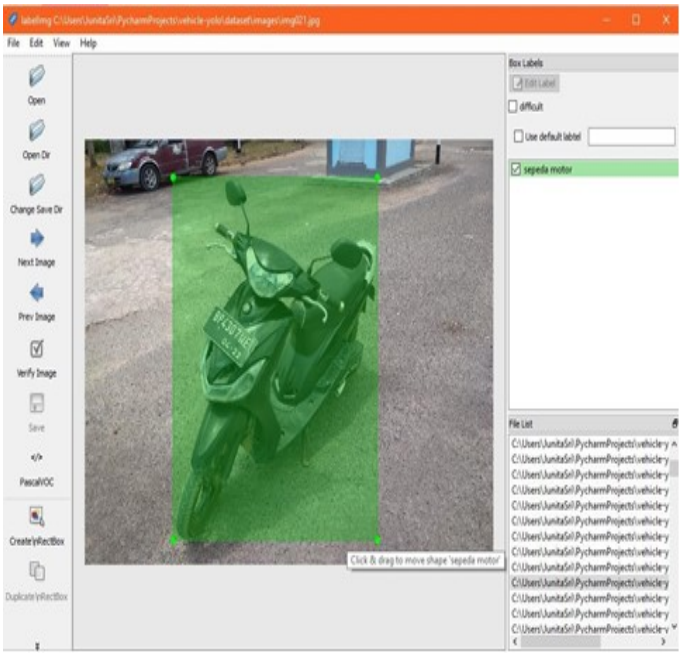
$$O = 1 + \frac{N + 2P - F}{S} \qquad (4)$$

O **:** ukuran *output* yang didapat setelah konvolusi.
N **:** ukuran *input*.
P **:** jumlah *padding*.
F **:** ukuran *kernel filter*.
S **:** jumlah *stride*.

C. Proses Training Sistem

Dalam penunjang penelitian ini, bahan yang digunakan untuk penelitian adalah data sample citra kendaraan yang terdapat di jalan raya sebagai sample pelatihan (*training*) dan hasil pengujian (*testing*) berupa video *real-time*. Tahap awal proses *training* adalah memberikan pelabelan dan *ground truth box* pada citra dataset menggunakan *labelImg*, yang akan disimpan dalam bentuk anotasi berformat *xml*. Penamaan label disesuaikan dengan kelas yang menjadi pembahasan pada penelitian ini, yaitu sepeda motor, mobil, truk, dan bus. Tampilan proses pemberian label pada dataset ditunjukkan pada Gambar 2.

Proses pelatihan dengan jumlah dataset sebanyak 200, 4 kelas, 10 *batch*, dan 200 *epoch*. *Batch* adalah jumlah pembagian dataset untuk setiap 1 langkah (*step*), sedangkan *epoch* adalah jumlah penggunaan keseluruhan dataset dalam tahap pelatihan. Dengan 200 dataset dan 10 *batch*, maka untuk menyelesaikan 1 *epoch* membutuhkan 200 / 10 = 20 *step* (langkah). Sehingga untuk menyelesaikan 200 *epoch*, membutuhkan 20 x 200 = 4.000 *step* (langkah). Setelah melewati 4.000 *step*, model yang telah dilatih (*trained model*) akan disimpan dalam bentuk *protobuf file* (*.pb) untuk kebutuhan pengujian pada perangkat *mobile*.



Gambar 2. Proses Pemberian Label Pada Dataset

III. HASIL DAN PEMBAHASAN

Proses pelatihan yang telah dilakukan hingga 4.000 *step*, dan penyimpanan *checkpoint* ke bentuk protobuf file dilakukan pada step 800, 1.000, 1.200, 1.400, 1.600, 1.800, 2.000, 3.000, dan 4.000. Setelah proses pelatihan, keseluruhan *protobuf file* ini diuji menggunakan gambar yang sama pada perangkat laptop. Pengujian pada perangkat laptop untuk gambar yang sama menunjukkan hasil yang dapat dilihat pada Tabel 1.

Tabel 1. Hasil Pengujian Pada Laptop

No	Jumlah Step	Jumlah Box	Jumlah Objek Terdeteksi	Akurasi	Nilai Keyakinan
1	800	9	3/6	5/10	0.1 ~ 0.206
2	1000	6	4/6	4/6	0.113 ~ 0.21
3	1200	6	5/6	5/6	0.119 ~ 0.23
4	1400	2	2/6	2/2	0.13 ~ 0.147

5	1600	1	2/6	1/1	0.143
6	1800	0	0	0	0
7	2000	0	0	0	0
8	3000	0	0	0	0
9	4000	0	0	0	0

Hasil tersebut menunjukkan bahwa *bounding box* yang muncul dan objek yang dapat terdeteksi semakin menurun seiring besarnya jumlah *step*, namun nilai keyakinannya sangat bervariasi untuk masing-masing *step*. Hasil ini menunjukkan bahwa kenaikan jumlah *step* tidak menentukan tingginya nilai keyakinan yang didapat. Kenaikan jumlah *step* hanya memberikan pengaruh pada penurunan jumlah *bounding box* sehingga tidak ada *box* yang saling menimpa pada objek yang sama.

Tabel 2. Deskripsi Hasil Pengujian Pada Perangkat mobile

No	Jumlah Step	Deskripsi Hasil Pengujian
1	800	Menghasilkan terlalu banyak *bounding box* yang tidak relevan terhadap objek, secara cepat.
2	1000	Frekuensi kemunculan *bounding box* yang tidak relevan terhadap objek masih lumayan tinggi, dengan waktu *delay* yang cukup cepat.
3	1200	Frekuensi kemunculan *bounding box* yang relevan terhadap objek, waktu *delay* relatif lambat terhadap pendeteksian.
4	1400	Cukup jarang memunculkan *bounding box* dan memiliki waktu *delay* yang relatif lambat terhadap pendeteksian.

Berdasarkan hasil pengujian tersebut, *protobuf file* yang akan dibawa untuk diuji di perangkat *mobile platform* android hanya *step* 800, 1.000, 1.200, dan 1.400. Keempat *trained* model ini diuji satu persatu pada perangkat *mobile*, untuk melihat performanya dapat dilihat pada Tabel 2.

Berdasarkan pengujian dengan empat trained model tersebut, hasil yang didapat menunjukkan bahwa jumlah step berpengaruh terhadap penurunan frekuensi kemunculan bounding box yang tidak relevan dengan objek.

Semakin tinggi jumlah *step*, semakin berkurang jumlah *bounding box* yang mendeteksi jalan raya sebagai objek kendaraan, dalam hal ini dapat disebut dengan *bounding box* asal-asalan. Kenaikan jumlah *step* juga memberikan dampak pada semakin melambatnya waktu *delay* untuk proses pendeteksian. Seringkali ditemui kondisi dimana *bounding box* muncul ± 2 detik setelah objek muncul. Pada kasus objek bergerak, *bounding box* mendeteksi objek tersebut secara benar namun kemunculan *bounding box* tersebut mengalami *delay*, dimana setelah objek meninggalkan *frame* kamera sekian detik kemudian *bounding box* baru muncul, sehingga terlihat tidak tepat pendeteksiannya. Dari keempat *trained model* yang diuji, *step* 1.200 merupakan *trained model* yang paling stabil dibandingkan 3 *trained model* lainnya.

Tabel 3. Hasil Perhitungan Tingkat Akurasi

No	Jumlah Step	Jumlah Objek Terdeteksi	Akurasi
1	800	3/6	50%
2	1000	4/6	66.6%
3	**1200**	**5/6**	**83.3%**
4	1400	2/6	33.3%
5	1600	2/6	33.3%
6	1800	0	0
7	2000	0	0
8	3000	0	0
9	4000	0	0

Perhitungan nilai akurasi bertujuan untuk mengetahui seberapa besar tingkat akurasi dari

ISSN 2615-6334 (Online)
ISSN 2087-5347 (Print)

metode YOLO yang digunakan dalam penelitian ini dapat dilihat pada Tabel 3.

Hasil tersebut menunjukkan bahwa dari perhitungan akurasi yang telah didapat, membuktikkan bahwa *step* 1.200 nilai akurasinya paling besar diantara *step* lainnya. *Step* 1.200 mendapatkan nilai akurasi sebesar **83.3%**, ini membuktikan bahwa *step* 1.200 merupakan *trained model* yang paling stabil dan mengklasifikasi objek secara tepat.

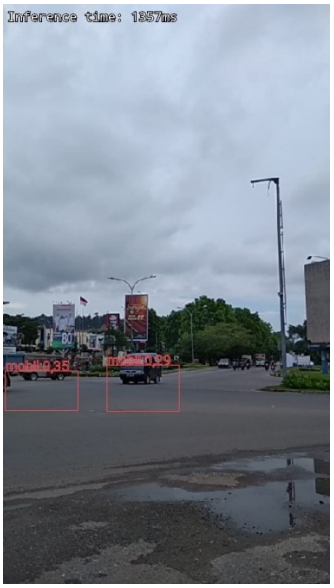
Gambar 3. *Bounding box* berhasil mendeteksi objek

Sedangkan *step* 1.000 menunjukkan bahwa nilai akurasi yang didapat sebesar 66.6% cukup stabil dalam mengklasifikasi objek secara tepat. Beberapa hasil uji melalui aplikasi android dengan menggunakan *trained model* 1200 ditunjukkan pada Gambar 3, Gambar 4, Gambar 5, dan Gambar 6.

Gambar 4. *Bounding box* berhasil mendeteksi objek

Pada Gambar 3 dapat dilihat bahwa hasil uji yang dilakukan secara langsung diperoleh bahwa *bounding box* mendeteksi beberapa objek secara akurat dengan *inference time* 1455 ms. Pada pengujian di android berhasil menampilkan 3 *bounding box* dalam waktu bersamaan.

Pada Gambar 4 dapat dilihat bahwa hasil uji yang dilakukan bahwa *bounding box* mendeteksi beberapa objek secara akurat dengan *inference time* 1377 ms. Pada pengujian di android berhasil menampilkan 2 *bounding box* dalam waktu bersamaan dengan jarak yang cukup jauh antara kendaraan dan tempat pengujian.

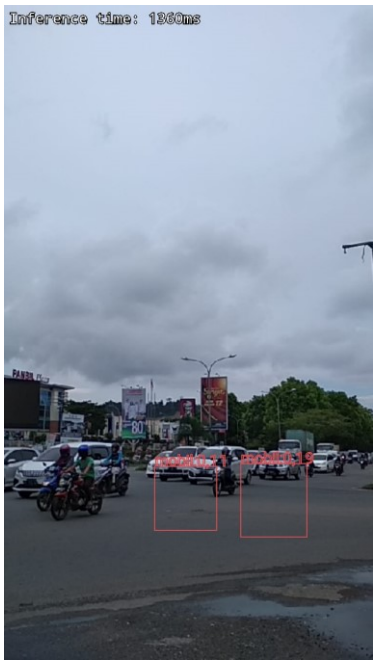


Gambar 5. *Bounding box* berhasil mendeteksi objek

Pada Gambar 5 diperoleh bahwa *bounding box* mendeteksi beberapa objek secara akurat dengan *inference time* 1357 ms. Pada pengujian di android berhasil menampilkan 2 *bounding box* dalam waktu bersamaan dan mengklasifikasi objek secara tepat.

Pada Gambar 6 dapat dilihat bahwa bahwa hasil uji yang dilakukan secara langsung diperoleh bahwa bounding box mendeteksi beberapa objek secara akurat dengan *inference* time 1360 ms. Pada pengujian di android berhasil menampilkan 2 *bounding box* dalam waktu bersamaan dan mengklasifikasi objek secara tepat namun terdapat *delay* meskipun objek yang terdeteksi tepat. Pengujian ini dilakukan menggunakan perangkat *smartphone* Xiaomi Redmi 4X dengan versi Android N (7.1.2) yang dibekali kemampuan *processor* inti

8 berkekuatan 1.4 GHz dan RAM sebesar 2 GB, dengan resolusi video berukuran 768x432 piksel.



Gambar 6. *Bounding box* berhasil mendeteksi objek

IV. KESIMPULAN

Pendeteksian kendaraan dengan menerapkan metode YOLO (*You Only Look Once*) memiliki performance yang dapat mendeteksi kendaraan sesuai dengan varian kendaraan yang telah diuji yaitu sepeda motor, mobil, truk dan bus. Dalam hal ini, berdasarkan pengujian yang telah dilakukan step 1200 merupakan *trained* model yang paling stabil dalam mengklasifikasi objek dengan menunjukkan nilai akurasi pendeteksian sebesar **83.3%**. Terbatasnya jumlah data latih dan resolusi kamera hp, juga mempengaruhi nilai keyakinan yang diperoleh pada Aplikasi Deteksi Kendaraan. Aplikasi Deteksi Kendaraan secara *real-time* dapat menampilkan *bounding box*, label kendaraan dan nilai keyakinan.

REFERENSI

[1] Redmon, J., Divvala, S., Girshick, R., dan Farhadi, A., 2015, "*You Only Look Once: Unified, Real-Time Object Detection*", Diterima dari http://arxiv.org/abs/1506.02640

[2] Serrano, A.S.I., 2017, "*YOLO Object Detector for Onboard Driving*", Universitat Autonoma de Barcelona

[3] Putra, M.H., Yussof, Z.M., Salim, S.I., dan Lim, K.C., 2017, "*Convolutional Neural Network for Person Detection using YOLO Framework*", *Journal of Telecommunication Electronic and Computer Engineering*, Vol. 9, No.(2), Hal. 9-13.

[4] 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

[5] Shafiee, M.J., Chywl, B., Li, F., dan Wong, A., 2017, "*Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video*", Diterima dari http://arxiv.org/abs/1709.05943.