Deteksi Jenis Mobil Menggunakan Metode YOLO Dan Faster R-CNN

Kevin Adiputra Shianto, Kartika Gunadi, Endang Setyati
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: kvndptr@gmail.com, kgunadi@petra.ac.id, endang@stts.edu

ABSTRAK
Jenis mobil merupakan salah satu properti dari mobil yang penting untuk diidentifikasi. Untuk melakukan identifikasi secara otomatis, banyak cara yang telah diimplementasi untuk mencapai tujuan identifikasi jenis mobil secara cepat dan tepat. Untuk identifikasi gambar, salah satu metode yang terkenal adalah Faster R-CNN yang cukup cepat dan tepat untuk melakukan identifikasi gambar. Namun untuk ketepatannya masih belum maksimal. Metode lain yang tersedia adalah YOLO dimana metode ini akan lebih cepat dalam melakukan identifikasi.

Penggunaan kedua metode dalam arsitektur yang akan dibangun bertujuan untuk meningkatkan akurasi kebenaran identifikasi jenis mobil. Kedua metode diharapkan dapat saling membantu dalam pengecekan dan menghasilkan hasil yang lebih baik dari masing-masing metode.

Kata Kunci: *Artificial Neural Network, Convolutional Neural Network, YOLO, Faster R-CNN, Image Recognition*

ABSTRACT
Car types is one of the properties from car to be identified. To identify automatically, many ways have been implemented to achieve the goal of identifying the type of car quickly and precisely. For image identification, one of the well-known methods is Faster R-CNN, which is fast and precise enough to identify images. But the accuracy is still not maximum. Another method available is YOLO where this method will be faster in identifying.

The use of both methods in the architecture to be built aims to improve the accuracy of the correct identification of the type of car. Both methods are expected to help each other in checking and produce better results from each method.

Keywords: *Artificial Neural Network, Convolutional Neural Network, YOLO, Faster R-CNN, Image Recognition*

1. PENDAHULUAN
Deteksi jenis mobil dapat berguna bagi keperluan lalu lintas, seperti melacak jenis mobil tertentu atau sistem pembayaran otomatis untuk jalan tol. Salah satu metode untuk melakukan *object detection* adalah Faster R-CNN. Namun Faster R-CNN masih belum maksimal dalam melakukan deteksi. Metode lain yang cukup baik dalam melakukan deteksi objek adalah YOLO.

Sistem yang akan dibuat menggabungkan hasil prediksi dari YOLO dan Faster R-CNN menggunakan sebuah *feed forward neural network*. Hasil dari *neural network* akan menggunakan data hasil YOLO dan Faster-RCNN sebagai inputnya lalu mengeluarkan sebuah *region* sebagai outputnya.

2. DASAR TEORI
2.1 Jenis Mobil
Jenis mobil yang akan dideteksi adalah sedan, *sport utility vehicle* (SUV), *multi purpose vehicle* (MPV), dan bus. Contoh mobil sedan dapat dilihat pada Gambar 1. Contoh mobil MPV dapat dilihat pada Gambar 2. Contoh mobil SUV dapat dilihat pada Gambar 3. Dan contoh mobil bus dapat dilihat pada Gambar 4.

Gambar 1. Contoh mobil sedan [14]

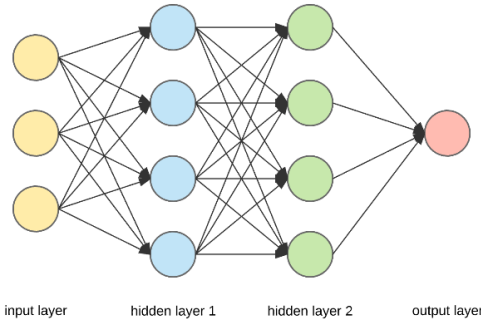
Gambar 2. Contoh mobil MPV [3]

Gambar 3. Contoh mobil SUV [3]

Gambar 4. Contoh mobil bus [4]

2.2 Artificial Neural Network

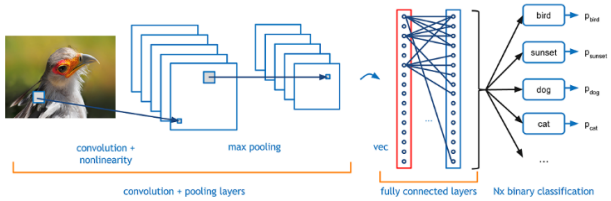
Artificial Neural Network atau yang dikenal dengan nama Jaringan Syaraf Tiruan (JST) merupakan sistem proses komputasi yang terinspirasi dari sistem saraf biologis manusia seperti otak. JST terdiri dari sejumlah interkoneksi node komputasi yang disebut *neuron*. Struktur dasar dari JST terdiri dari 3 jenis layer utama, yaitu *input*¸ *hidden*, dan *output* layer [13]. Ilustrasi model dasar jaringan syaraf tiruan dapat dilihat pada Gambar 5.



Gambar 5. Model dasar jaringan syaraf tiruan [8]

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan JST yang *neuron* di dalam *layer*-nya tersusun menjadi 3 dimensi [13]. Berbeda dengan JST pada umumnya, *neuron* dalam *layer* tertentu pada CNN hanya akan terhubung pada ke wilayah kecil pada *layer* sebelumnya. O'Shea dan Nash juga menjelaskan bahwa penggunaan CNN difokuskan pada neural network yang menerima input berupa citra (*image*). Ilustrasi arsitektur CNN dapat dilihat pada Gambar 6.



Gambar 6. Arsitektur Convolutional Neural Network [8]

2.4 Faster Region-based Convolutional Neural Network

Faster Region-based Convolutional Neural Network atau biasa disingkat Faster R-CNN adalah sebuah metode yang digunakan untuk mendeteksi objek pada sebuah gambar. Faster R-CNN adalah metode yang menggunakan Fast R-CNN dan RPN (Region Proposal Network) sebagai arsitektur utamanya. Metode ini sama dengan Fast R-CNN kecuali bagian selective search pada Fast R-CNN diganti dengan RPN [11].

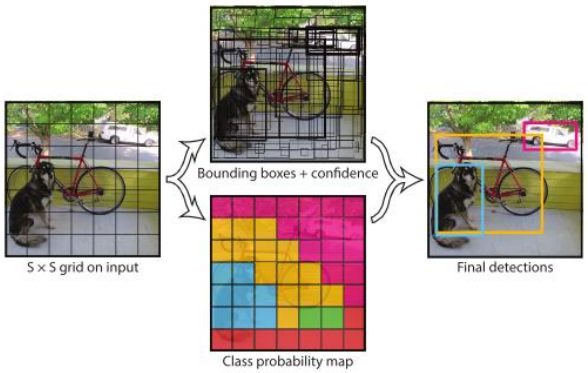
Langkah-langkah dalam Faster R-CNN adalah:

1. Konvolusi (Pengambilan *Feature Map*) [11]
2. *Region Proposal Network* (RPN) [5]
3. ROI Pooling [6]
4. *Convolutional Neural Network* [7]

2.5 You Only Look Once

You Only Look Once atau YOLO akan membagi inputan gambar menjadi *grid* berukuran S×S, dimana nilai S adalah 7 dengan input gambar berukuran 448 x 448. Untuk mendapatkan *bounding box*, akan dilakukan konvolusi dari inputan gambar, sehingga hasil akhirnya akan mendapat ukuran bounding box sebesar S×S×(B*5+C) dimana B adalah banyaknya bounding box (umumnya 2) dalam 1 grid dan C adalah banyaknya *class* yang dapat diklasifikasi. Nilai B dikalikan dengan 5 karena sebuah bounding box memiliki 5 nilai yang perlu disimpan, koordinat x, koordinat y, lebar (width), tinggi (height), dan confidence score (nilai probabilitas bounding box yang bersangkutan memiliki sebuah objek) [10].

Untuk semua atribut pada bounding box akan dilakukan normalisasi sehingga nilainya menjadi antara 0 hingga 1. Koordinat x dan y akan dinormalisasi menyesuaikan titik kiri atas dari grid yang bersangkutan. Dan tinggi dan lebar akan dinormalisasi sesuai dengan ukuran gambar (width dan height). Nilai koordinat x dan y pada sebuah bounding box pada setiap grid merupakan titik tengah grid yang bersangkutan [8]. Ilustrasi YOLO dapat dilihat pada Gambar 7.

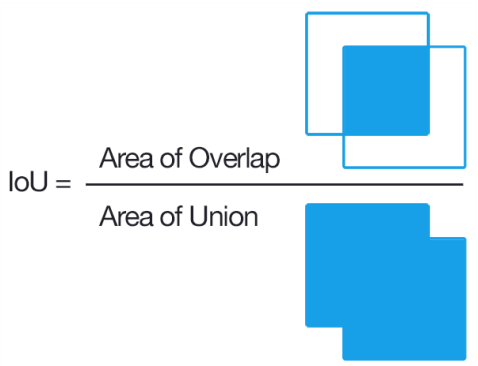


Gambar 7. You Only Look Once [10]

2.6 Intersection Over Union

Intersection over Union (IoU) merupakan metode evaluasi untuk mengukur akurasi deteksi objek terhadap suatu *dataset*. IoU membutuhkan 2 area yang akan di-*intersect* dan di-*union*, 2 area tersebut adalah area *ground-truth bounding box* yang merupakan *bounding box* aktual dan area yang dideteksi dari model yang

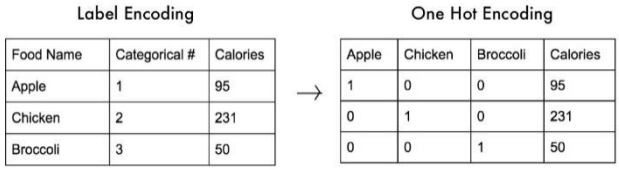
dibangun [12]. Ilustrasi persamaan IoU dapat dilihat pada Gambar 8.



Gambar 8. Intersection over Union [12]

2.7 One Hot Encoding

One Hot Encoding adalah cara merepresentasikan data yang digunakan agar dapat dimenegerti oleh komputer. Cara kerja *One Hot* adalah dengan membuat sebuah *tensor* atau *array* 1 dimensi dengan panjang sebanyak jenis *class* yang ada dan mempunyai isi biner antara 0 atau 1 [2]. Ilustasi *One Hot Encoding* dapat dilihat pada Gambar 9.



Gambar 9. Contoh One Hot Encoding [2]

2.8 Confusion Matrix

Confusion matrix adalah sebuah tabel yang menunjukkan kinerja dari sebuah model klasifikasi yang memiliki data jawaban benar (*supervise*). Dari tabel yang didapatkan, untuk model klasifikasi yang dimiliki dapat dihitung akurasi, presisi, F-Score dan masih banyak lagi variabel yang dapat dihitung berdasarkan kondisi data yang diprediksi atau diklasifikasikan [1].

Isi dari tabel confusion matrix ada 4, yaitu

- True Positive (TP), kondisi dimana model mengklasifikasikan data sebagai ya (TRUE) dan jawaban aktualnya adalah ya (TRUE)
- True Negative (TN), kondisi dimana model mengklasifikasikan data sebagai tidak (FALSE) dan jawaban aktualnya adalah tidak (FALSE)
- False Positive (FP), kondisi dimana model mengklasifikasikan data sebagai ya (TRUE) dan jawaban aktualnya adalah tidak (FALSE)
- False Negative (FN), kondisi dimana model mengklasifikasikan data sebagai tidak (FALSE) dan jawaban aktualnya adalah ya (TRUE)

Persamaan yang akan digunakan berdasarkan data dari *confusion matrix* adalah akurasi, presisi, *recall* dan *f-score.* Akurasi merupakan pengukuran seberapa benar sebuah sistem dapat mengklasifikasi dari keseluruhan. Akurasi dapat dihitung menggunakan persamaan (1).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

Presisi merupakan perbandingan jumlah data yang kategori positif yang diklasifikasikan secara benar oleh sistem dan keseluruhan data yang terklasifikasi positif. Presisi dapat dihitung menggunakan persamaan (2).

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall merupakan pengukuran untuk data dengan klasifikasi positif yang benar oleh sistem. *Recall* dapat dihitung menggunakan persamaan (3).

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

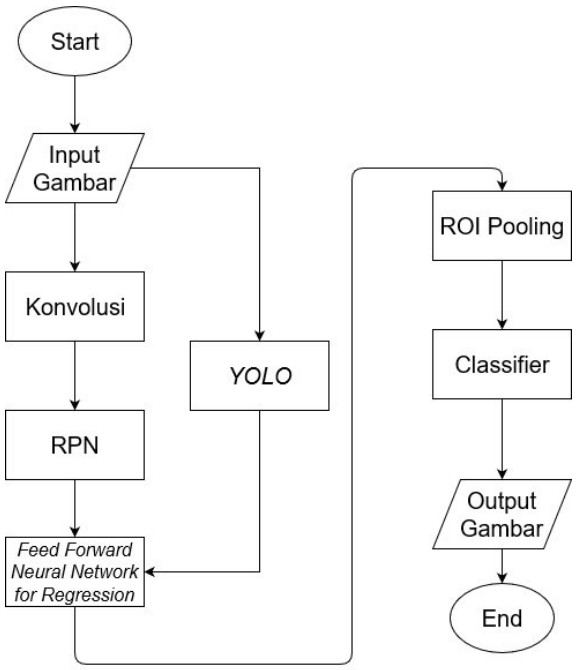
F-Score bertujuan untuk menghitung kombinasi dari presisi dan *recall*. *F-Score* akan menggunakan *harmonic mean* dari presisi dan *recall*. *F-score* dapat dihitung menggunakan persamaan (4).

$$F\text{-}Score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (4)$$

3. DESAIN SISTEM

3.1 Analisis Sistem

Analisis sistem membahas permasalahan bagaimana data yang diinputkan diproses sehingga dapat menghasilkan output yang sesuai. Sistem mencakup pengambilan fitur, model *neural network* untuk *training*, dan sistem untuk *testing*. Desain arsitektur *neural network* yang akan dibentuk dapat dilihat pada diagram Gambar 10.



Gambar 10. Desain arsitektur *neural* network

3.1.1 Training Neural Network

Pada model *neural network* semua *weight* diinisialisasi menggunakan *random weight* mengikuti *normal distribution*. Terdapat 3 (tiga) bagian network utama, yaitu RPN dan CNN untuk Faster R-CNN, dan YOLO. Ketiga *neural network* akan melakukan *training* secara terpisah. Konfigurasi awal seperti jumlah *epoch*,

nilai *learning rate* dan jumlah *batch* akan disamakan. Dataset yang digunakan untuk ketiga *neural network* juga sama.

3.1.1.1 RPN

RPN akan menerima input berupa *feature map* yang telah diproses oleh konvolusi lalu menjalankan *sliding window* untuk menghasilkan proposal berdasarkan *anchor* yang telah ditentukan. Untuk setiap proposal akan dilakukan konvolusi untuk menghasilkan nilai objektivitas (0-1) dimana nilai tersebut menandakan keberadaan objek atau *foreground*.

Proposal-proposal yang dihasilkan oleh RPN kemudian dibandingkan dengan *Ground Truth Box* menggunakan *Intersection Over Union*. Jika hasil dari IoU lebih kecil dari *threshold negative overlap* (default: 0.3) maka proposal tersebut akan dianggap *background* dan tidak akan lanjut ke tahap berikutnya. Selain itu proposal akan dianggap mengandung *foreground* atau objek [9].

3.1.1.2 CNN

Network terakhir pada Faster R-CNN yang digunakan untuk klasifikasi dan regresi untuk menentukan lokasi atau *region* yang mengandung objek pada sebuah gambar dan *class* dari objek tersebut adalah CNN. Setelah mendapatkan *region* yang mengandung objek, *region-region* tersebut akan melalui tahap yang dinamakan ROI Pooling. ROI Pooling berfungsi untuk menyamakan dimensi *width* dan *height* dari *region-region* yang telah didapatkan oleh RPN. Hal ini dilakukan karena CNN hanya dapat menerima input dengan dimensi yang sama.

ROI Pooling akan menyamakan dimensi region-region proposal yang didapatkan menjadi 56 x 56. Ukuran 56 karena CNN yang digunakan adalah VGGNET dengan jumlah layer yang dikurangi sehingga input yang awalnya adalah 224 x 224 menjadi 56 x 56. *Activation function* yang digunakan adalah ReLU. *Loss* function yang digunakan adalah *cross entropy*.

3.1.1.3 YOLO

Neural Network YOLO mirip dengan model CNN. Fitur yang diberikan kepada YOLO akan melalui 24 konvolusi, 4 *max pooling* dan 2 *fully connected layer* untuk mendapatkan *grid* yang mengandung nilai untuk diklasifikasi dan diregresi. *Activation function* yang digunakan pada *layer* terakhir adalah *linear activation function*, sedangkan pada *layer* lainnya menggunakan *leaky ReLU*.

Setelah mendapatkan *grid-grid* yang dihasilkan dari konvolusi, *classification* akan dilakukan diawali dengan melakukan perkalian antara *confidence score* tiap *bounding* box dengan semua *class* score pada *grid*-nya sehingga mendapatkan *S x S x* 2 tensor. Kemudian mengubah nilai *class* dari setiap *grid* yang lebih rendah dari nilai *threshold* yang telah ditentukan menjadi 0. NMS akan dilakukan untuk mencari *redundant boxes* (2 atau lebih *bounding box* yang mengandung 1 objek yang sama) dan kemudian *score* untuk *class* tersebut akan diubah menjadi 0. Dan nilai *class* tertinggi akan menjadi *class* pada *bounding box* tersebut.

3.1.2 Testing Neural Network

Testing akan dilakukan secara terstruktur, tidak terpisah seperti sewaktu *training*. *Testing* yang akan dilakukan terbagi menjadi 2 jenis sebagai perbandingan bagaimana pengaruh YOLO terhadap Faster R-CNN. Yang pertama adalah dari RPN langsung ke *classification & regression* (Faster R-CNN biasa) dan yang kedua adalah dari RPN menuju suatu proses yang menghitung output dari RPN dan YOLO untuk meningkatkan *proposal* yang dihasilkan, lalu menuju ke *classification* dan *regression*.

3.2 Desain Aplikasi

Aplikasi dibuat dengan menggunakan php. Aplikasi yang dibuat memiliki fitur untuk mengupload sebuah gambar yang akan dideteksi jenis mobil dan lokasi mobilnya pada gambar yang diupload. Setelah *user* mengupload gambar yang berisikan mobil yang ingin dideteksi maka tampilan web akan menunjukkan gambar awal, gambar dengan *bounding box* yang diprediksi dan gambar yang di-*crop* sesuai *bounding box*-nya, dan hasil klasifikasi untuk metode Faster R-CNN, YOLO dan gabungan antara Faster R-CNN dan YOLO.

4. IMPLEMENTASI SISTEM

Implementasi sistem dilakukan pada computer dengan spesifikasi RAM: 8GB DDR3, GPU: Nvidia GeForce GTX 860M, HDD: 1 TB, CPU: Core i7, dan OS Windows 10 Professional.

Implementasi pengkodean sistem, menggunakan Bahasa pemrograman Python dengan versi 3.5.0. *Framework* yang digunakan untuk sistem ini adalah Tensorflow. Adapun beberapa *library* yang mendukung sistem ini adalah Matplotlib, CV2, dan Numpy.

Selain itu aplikasi web pada php akan menjalankan file python menggunakan *command exec* yang ada pada php untuk melakukan prediksi pada gambar yang di-*upload*.

5. ANALISA DAN PENGUJIAN

5.1 Pengujian Sistem

Pengujian sistem mengevaluasi performa daripada sistem *training* dan sistem *testing* yang telah dibuat. Pencatatan pengujian akan dibedakan berdasarkan jenis mobil yang diprediksi dan diklasifikasi. Prediksi *region* akan menggunakan metode perhitungan IoU yang dirata-ratakan sementara klasifikasi akan menggunakan *confusion matrix*.

5.1.1 Pengujian Metode YOLO

Pengujian dengan menggunakan metode YOLO dilakukan berdasarkan 4 tahap *epoch* yang berbeda. Pengujian dilakukan dengan *epoch* 25, 50, 75, dan 100. Hasil pengujian dapat dilihat pada Tabel 1 dan Tabel 2.

Tabel 1. Akurasi Hasil Testing Prediksi *Bounding Box* YOLO

Epoch	Sedan	SUV	MPV	Bus
25	6,546%	19,008%	5,853%	16,095%
50	**36,394%**	**55,618%**	**34,394%**	**32,017%**
75	31,650%	45,794%	20,943%	12,628%
100	41,047%	54,140%	33,323%	24,051%

Tabel 2 Akurasi Hasil Testing Klasifikasi YOLO

Epoch	Sedan	SUV	MPV	Bus
25	13,416%	13,479%	13,228%	13,793%
50	**43,887%**	**43,887%**	**42,946%**	**45,329%**
75	29,717%	30,407%	29,843%	31,159%
100	42,006%	42,319%	41,818%	42,633%

5.1.2 Pengujian Metode Faster R-CNN

Pengujian pada metode Faster R-CNN terdiri dari pengujian prediksi *bounding box* (menggunakan RPN) dan pengujian klasifikasi (menggunakan CNN). Prediksi dan klasifikasi akan menggunakan *epoch* yang sama, yaitu 25, 50, 75, dan 100. Hasil pengujian dapat dilihat pada Tabel 3 dan Tabel 4.

Tabel 3. Akurasi Hasil Testing Prediksi *Bounding Box* RPN

Epoch	Sedan	SUV	MPV	Bus
25	47,431%	48,588%	45,005%	36,330%
50	53,666%	53,653%	49,587%	43,454%
75	57,382%	56,891%	54,092%	46,380%
100	**61,329%**	**60,733%**	**68,515%**	**49,529%**

Tabel 4. Akurasi Hasil Testing Prediksi RPN

Epoch	Sedan	SUV	MPV	Bus
25	66,081%	71,097%	66,896%	76,426%
50	71,724%	71,473%	71,222%	75,360%
75	72,727%	74,545%	73,542%	80,501%
100	**73,542%**	**74,231%**	**72,476%**	**81,191%**

5.1.3 Pengujian Metode Faster R-CNN dan YOLO

Pengujian pada metode Faster R-CNN dan YOLO terdiri dari pengujian prediksi yang menggunakan sebuah *neural network* untuk menghitung *weight* dari YOLO dan Faster R-CNN dan pengujian klasifikasi (menggunakan CNN). Prediksi akan menggunakan epoch 250, 500, 750, dan 1000. Sementara untuk klasifikasi akan menggunakan epoch 25, 50, 75, dan 100. Hasil YOLO yang akan digunakan diambil dari epoch 50 dimana hasil maksimum untuk pengujian YOLO jatuh pada *epoch* ke-50. Hasil pengujian dapat dilihat pada Tabel 5 dan Tabel 6.

Tabel 5. Akurasi Hasil Testing Prediksi *Bounding* Box RPN dan YOLO

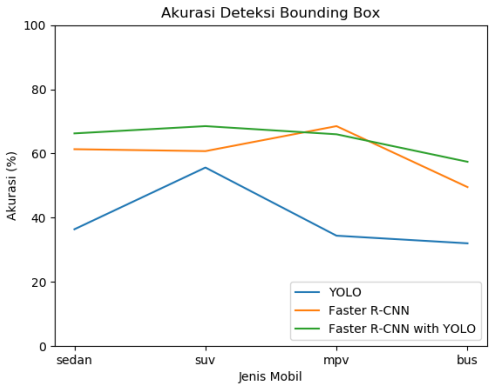
Epoch	Sedan	SUV	MPV	Bus
50	65,516%	68,021%	65,006%	56,776%
100	66,029%	68,360%	65,738%	57,053%
150	66,141%	68,421%	65,883%	57,200%
200	66,174%	68,462%	65,910%	57,271%
250	66,208%	68,480%	65,939%	57,318%
300	66,222%	68,502%	65,950%	57,349%
350	66,235%	68,511%	65,962%	57,368%
400	66,238%	68,510%	65,967%	57,379%
450	66,243%	68,516%	65,971%	57,385%
500	66,252%	68,518%	65,971%	57,395%
550	66,250%	68,513%	65,972%	57,397%
600	66,251%	68,513%	65,972%	57,400%
650	66,247%	68,513%	65,978%	57,404%
700	66,252%	68,513%	65,981%	57,405%
750	66,252%	68,513%	65,981%	57,405%
800	66,253%	68,515%	65,979%	57,406%
850	66,255%	68,514%	65,979%	57,410%
900	**66,255%**	**68,516%**	**65,978%**	**57,411%**
950	66,255%	68,515%	65,979%	57,411%
1000	66,255%	68,515%	65,979%	57,411%

Tabel 6. Akurasi Hasil Testing Klasifikasi CNN

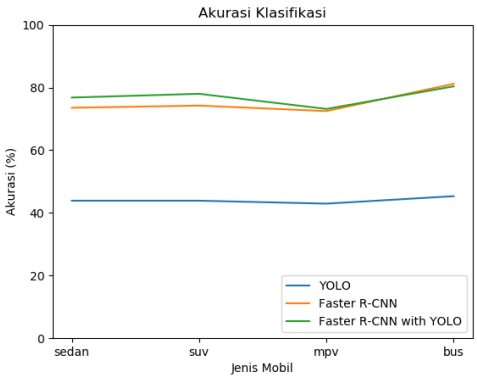
Epoch	Sedan	SUV	MPV	Bus
25	73,730%	78,871%	69,780%	82,570%
50	75,611%	78,369%	73,166%	77,304%
75	**76,802%**	**77,993%**	**73,166%**	**80,376%**
100	75,297%	76,614%	72,978%	80,940%

5.1.4 Perbandingan Metode YOLO, Faster R-CNN, dan Faster R-CNN dengan YOLO

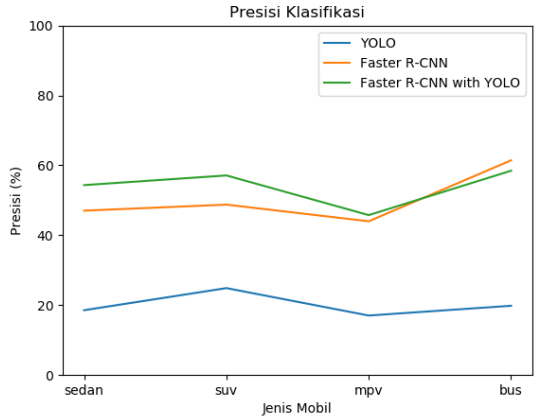
Pada pengujian sistem untuk ketiga metode yang dilakukan telah ditemukan *epoch* terbaik yang menghasilkan akurasi tertinggi pada setiap *epoch*-nya. Untuk setiap metode akan diambil *epoch* terbaiknya lalu dibandingkan satu sama lain. Perbandingan akurasi deteksi *bounding box* dapat dilihat pada Gambar 11. Perbandingan akurasi klasifikasi dapat dilihat pada Gambar 12. Perbandingan presisi klasifikasi dapat dilihat pada Gambar 13. Perbandingan *recall* klasifikasi dapat dilihat pada Gambar 14. Dan perbandingan *f-score* dapat dilihat pada Gambar 15.



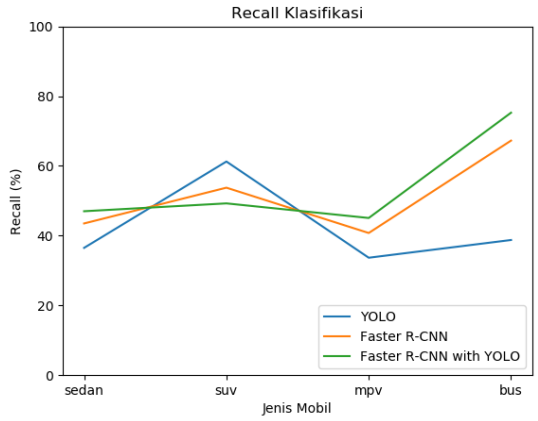
Gambar 11. Grafik perbandingan akurasi deteksi *bounding box* metode YOLO, Faster R-CNN, dan Faster R-CNN dengan YOLO



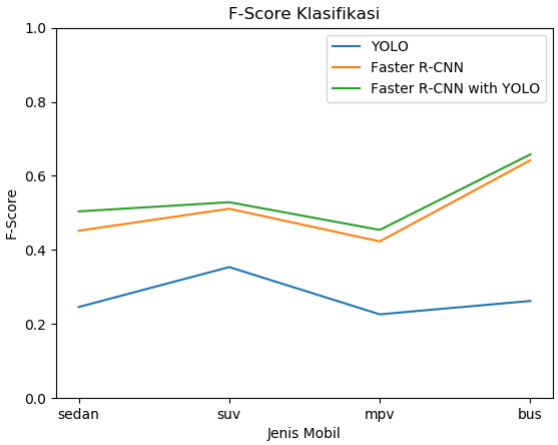
Gambar 12. Grafik perbandingan akurasi klasifikasi metode YOLO, Faster R-CNN, dan Faster R-CNN dengan YOLO



Gambar 13. Grafik perbandingan presisi klasifikasi metode YOLO, Faster R-CNN, dan Faster R-CNN dengan YOLO



Gambar 14. Grafik perbandingan *recall* klasifikasi metode YOLO, Faster R-CNN, dan Faster R-CNN dengan YOLO



Gambar 15. Grafik perbandingan *f-score* klasifikasi metode YOLO, Faster R-CNN, dan Faster R-CNN dengan YOLO

Deteksi *bounding box* untuk Faster R-CNN meningkat setelah digabungkan dengan YOLO menggunakan sebuah *neural network* yang dirancang khusus untuk memberikan *output* letak *bounding box* yang terbaik berdasarkan *bounding box* hasil Faster R-CNN dan YOLO. Untuk klasifikasi, akurasi Faster R-CNN juga meningkat setelah digabungkan dengan YOLO. CNN yang digunakan untuk klasifikasi menggunakan konfigurasi *layer* dan *weigth* yang sama.

5.2 Pengujian Aplikasi

Pengujian aplikasi dilakukan dengan mencoba melakukan prediksi dan klasifikasi berdasarkan *user interface* yang telah dibuat. Pengujian dilakukan pada masing-masing jenis mobil untuk melihat bagaimana hasil dari prediksi *bounding box* dan klasifikasi pada *user interface* yang telah dibuat. Contoh prediksi dan klasifikasi yang dilakukan dapat dilihat pada Gambar 16.

Gambar 16. Contoh *user interface* klasifikasi dan prediksi

6. KESIMPULAN

Setelah dilakukan perancangan sistem, pengimplementasian, dan pengujian terhadap aplikasi yang telah dibuat, dapat ditarik kesimpulan sebagai berikut:

- YOLO memiliki akurasi prediksi yang lebih rendah dari Faster R-CNN, namun dapat memprediksi lebih akurat dari Faster R-CNN.
- Akurasi prediksi YOLO lebih rendah dari Faster R-CNN karena YOLO lebih banyak tidak mendapatkan region saat melakukan prediksi dibandingkan dengan prediksi Faster-RCNN.
- Keakuratan dalam melakukan klasifikasi dipengaruhi oleh region yang berhasil diprediksi.
- Penambahan metode YOLO dapat menambah akurasi deteksi RPN.

Saran untuk pengembangan kedepannnya adalah:

- Menggunakan sistem server-client untuk aplikasi agar dari pihak pengguna aplikasi tidak perlu menjalankan tensorflow lagi.
- Penambahan dataset yang lebih bervariasi.
- Penambahan layer lain pada model CNN.

7. DAFTAR PUSTAKA

[1] DataSchool.io. 2014. *Simple guide to confusion matrix terminology*. Diambil kembali dari DataSchool: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[2] DelSole, M. *What is One Hot Encoding and How to Do It*. Diambil kembali dari Medium: https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179

[3] Dory, J. 2015. *What is the difference between SUV, MPV and Crossovers?*. Diambil kembali dari Medium: https://medium.com/@JohnDory/what-is-the-difference-between-suvs-mpvs-and-crossovers-37dd580fbffc

[4] Easton, A. H., & Cromer, G. C. *Bus Vehicle*. Diambil kembali dari Britannica: https://www.britannica.com/technology/bus-vehicle

[5] Gao, H. 2017. *Faster R-CNN Explained*. Diambil kembali dari: https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8

[6] Grel, T. 2017. *Region of interest pooling explained*. Diambil kembali dari DeepSense: https://blog.deepsense.ai/region-of-interest-pooling-explained/

[7] Javier. 2018. *Faster R-CNN: Down the rabbit hole of modern object detection.* Diambil kembali dari TryoLabs: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

[8] Menegaz, M. *Understanding YOLO – Hacker Noon*. Diambil kembali dari HackerNoon: https://hackernoon.com/understanding-yolo-f5a74bbc7967

[9] Mohan, A. 2018. *Object Detection and Classification Using R-CNNs*. Diambil kembali dari Telesens: http://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/

[10] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. *arXiv*, 1-10.

[11] Ren, S., He, K., Girshick, R., & Sun, J. 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv*, 1-14.

[12] Rosebrock, A. 2016. *Intersection over Union (IoU) for object detection.* Diambil kembali dari PYImageSearch: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[13] O'Shea, K., & Nash, R. 2015. *An Introduction to Convolutional Neural Networks*. Diambil kembali dari arXiv: https://arxiv.org/abs/1511.08458

[14] Shankar, S. 2015. *Different Car Body Types*. Diambil kembali dari CarTrade: https://www.cartrade.com/blog/2013/auto-guides/different-car-body-types-494.html