Deteksi Pelanggaran pada Bahu Jalan Tol Menggunakan Algoritma Cascade R-CNN

Belinda Fridolin Darmanto
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
belindadarmanto@student.telkomuniversity.ac.id

Casi Setianingsih
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
setiacasie@telkomuniversity.ac.id

Randy Erfa Saputra
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
resaputra@telkomuniversity.ac.id

Abstrak— Bahu jalan tol merupakan bagian kiri jalan tol yang berguna bagi kendaraan yang mengalami keadaan darurat. Namun masih banyak kendaran yang menggunakan bahu jalan tol dengan tidak bijak seperti menggunakannya untuk mengdahului kendaraan lain atau berhenti atau beristirahat dalam waktu yang lama. Hal ini dapat menyebabkan kecelakaan dan juga keributan.

Berdasarkan dari masalah tersebut, diaplikasikan algoritma *Cascade R-CNN* yang dapat digunakan sebagai algoritma untuk mendeteksi pelanggaran pada bahu jalan tol. *Cascade R-CNN* memiliki deteksi *multistage* yang dapat mengurangi *overfitting* karena kurangnya dataset. *Cascade R-CNN* memiliki deteksi *multistage* yang terdiri dari tiga *stage*. Hasil *train* deteksi di *stage* pertama akan di *train* lagi di *stage* kedua, begitu pun seterusnya sampai di hasil *stage* ketiga. Deteksi multi*stage* ini membuat *Cascade R-CNN* disebut sebagai algoritma berkualitas tinggi dalam mendeteksi objek.

Pengujian Algoritma *Cascade R-CNN* dilakukan menggunakan tiga *hyperparameter* yaitu *epoch*, *batch size*, dan *learning rate*. Pengujian *hyperparameter* ini bertujuan untuk mendapatkan model terbaik untuk melakukan prediksi. Model terbaik didapatkan pada *hyperparameter* di *epoch* 12, *batch size* 16, *learning rate* 0.02 dengan *Average Precision*=97,1%, *Average Recall*=79,1%, mAP@.5=97,1% , dan mAP@.5:.95=74,8%.

Kata kunci— : Bahu Jalan Tol, *Cascade R-CNN*, Deteksi Objek

I. PENDAHULUAN

Jalan tol adalah bagian dari suatu sistem jaringan jalanan umum yang dapat dilalui kendaraan seperti kendaraan roda empat, bus dan truk. Jalan tol sudah menjadi jalan alternatif untuk mempersingkat waktu perjalanan jika jalan umum sedang macet dan ingin cepat sampai tujuan. Pada jalan tol terdapat bahu jalan tol yang fungsinya adalah sebagai tempat berhenti jika megalami keadaan darurat saja dan jika berhenti pun kendaraan tidak boleh berhenti berlama-lama di bahu jalan tol. Fungsi bahu jalan tol juga adalah sebagai arus lalu lintas bagi keadaann darurat. Dalam Peraturan Pemerintah No. 15 Tahun 2005 tentang Jalan Tol bahwa bahu jalan tol tidak boleh digunakan untuk mendahului kendaraan yang lain[1].

Cascade R-CNN yang bisa mendeteksi pelanggaran pada bahu jalan tol yang bisa mendeteksi pada dataset sama atau berbeda. Algoritma ini sangat efektif karena ia dapat mengurangi *overfitting* karena kurangnya dataset. *Cascade R-CNN* memiliki deteksi multi*stage* yang terdiri dari tiga *stage*. Hasil *train* deteksi di *stage* pertama akan di *train* lagi di *stage* kedua, begitu pun seterusnya sampai di hasil *stage* ketiga[2]

.

Maka dari itu proposal ini diajukan untuk ingin membuat sistem deteksi pelanggaran pada bahu jalan tol. Sistem ini menggunakan algoritma *Cascade R-CNN* untuk mendeteksi kendaraan yang menggunakan bahu jalan tol untuk mendahului kendaraan lain atau berhenti terlalu lama di bahu jalan tol. .

Jalan tol merupakan jalanan jalur cepat yang umumnya dilalui oleh kendaraan seperti mobil, bus, dan truk, yang bertujuan untuk mempersingkat jarak perjalanan pengemudi kendaraan. Menurut Peraturan Pemerintah Nomor 15 Tahun 2005 Pasal 41 [1]. Pada jalan tol terdapat bahu jalan yang dipergunakan untuk pengemudi yang mengalami keadaan darurat saja, namun pengemudi yang tidak mengalami keadaan darurat tidak diperbolehkan untuk menggunakan bahu jalan.

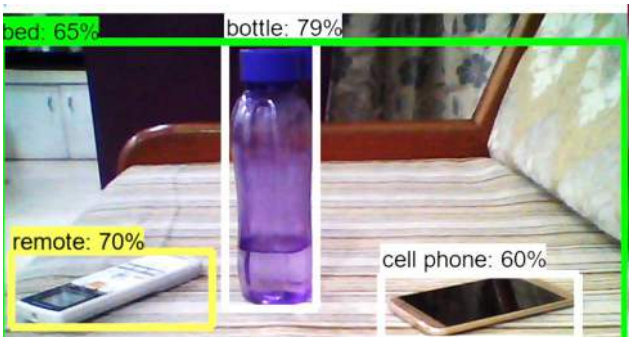
II. KAJIAN TEORI

A. Deteksi Objek

Deteksi objek merupakan deteksi beberapa *instances* dari beberapa objek atau kelas dalam gambar. Jika sebuah program memilliki kelas atau objek dengan nama Buah, maka *instances* dari Buah adalah jeruk, apel dan pisang. Tujuan deteksi objek adalah untuk mendeteksi semua *instances* dari suatu atau beberapa objek atau kelas seperti orang, mobil atau wajah dalam suatu gambar[3].

Setiap deteksi diketahui dari informasi *pose*. *Pose* adalah posisi dari suatu objek yang biasanya dalam bentuk tiga dimensi *pose* bisanaya disimpan sebagai matriks transformasi. *Pose* dari sebuah objek bisa diketahui dari input kamera dengan menggunakan proses dari *pose estimation*. *Pose estimation* adalah teknik dari *computer vision* yang memprediksi dan melacak lokasi dari sebuah objek[4].

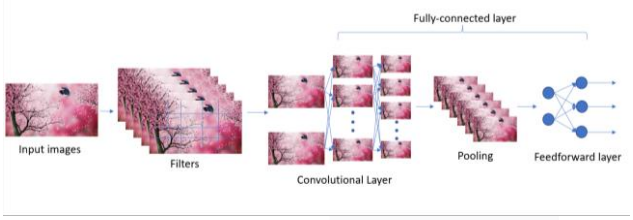
Dalam situasi yang berbeda, *pose information* bisa menjadi lebih detail dan juga bisa memiliki parameter transformasi *non-linear* atau *linear*. Misalnya detektor wajah dapat memberikan *bounding box* pada wajah namun *bounding box* tersebut juga bisa mendetksi bagian dari wajah seperti hidung, mata, dan juga bibir. *Pose* juga dapat didefinisikan oleh transformasi tiga dimensi yang menentukan lokasi objek relatif terhadap kamera[3], [5]Contohnya seperti gambar-gambar dibawah ini.



Gambar 1. Objeck Detection [6]

B. Convolutional Neural Network (C-NN)

Convolutional Neural Network adalah *neural network* yang menggabungkan dua set informasi menggunakan konvolusi operasi matematika. Konvolusi digunakan sebagi *filter* informasi menjadi sebuah fitur mAP. *Filter* dalam konvolusi disebut sebagai kernel berupa dimensi seperti 5x5 atau disebut perkalian matriks. Untuk melakukan konvolusi, kernel melewati *input* kemudian melakukan perkalian matriks tiap elemen yang hasilnya akan ditulis pada fitur mAP. Konvolosi pada gambar sebagian besar memiliki 3 dimensi tinggi, lebar dan kedalaman, dimana memiliki warna sesuai dengan kedalaman gambar yaitu (RGB). Dalam lapisan konvolusi memiliki beberapa *filter*, masing-masing menghasilkan mAP *filter*. *Output* dari sebuah layer akan menjadi satu mAP *filter* yang ditumpuk satu sama lain[7].

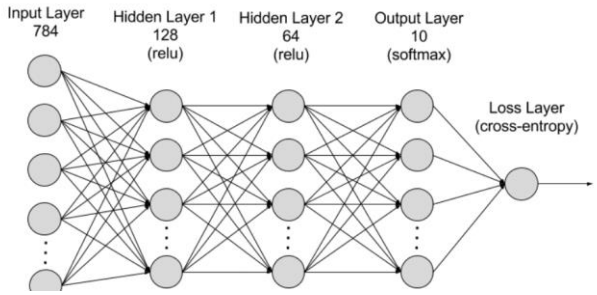


Gambar 2. Convolutional Neural Network [7]

Convolutional Neural Network memiliki convolutional layer, pooling layer dan fully-connected layer. Convolutional layer berguna untuk mengenali fitur dalam pixel. Pooling layer berguna dalam membuat fitur abstrak untuk mendapatkan pooling fitur mAP. Fully-connected layer berguna untuk memperoleh hasil prediksi yang dimana output yang telah diratakan dimasukkan kedalam feed-forward neural network dan backpropagation kemudian pada lapisan ini menyediakan model untuk memahami gambar [2].

C. Deep Learning

Deep learning adalah metode dalam kecerdasan buatan (AI) yang mengajarkan komputer untuk memproses data yang terinspirasi otak manusia. Model *deep learning* dapat mengenali pola kompleks dalam gambar, teks, suara, dan data lain untuk menghasilkan wawasan dan prediksi yang akurat. Algoritma *deep learning* merupakan jaringan neural yang meniru otak manusia. Misalnya, otak manusia memiliki jutaan neuron yang saling terhubung yang bekerja sama untuk mempelajari dan memproses informasi. Demikian pula, jaringan *neural deep learning*, atau jaringan neural buatan, terbuat dari banyak lapisan neuron buatan yang bekerja sama di dalam computer. Neuron buatan adalah modul perangkat lunak yang disebut simpul, yang menggunakan perhitungan matematika untuk memproses data. Jaringan neural buatan adalah algoritme *deep learning* yang menggunakan simpul ini untuk memecahkan masalah kompleks[8].



Gambar 3. Komponen Jaringan Deep Learning[8]

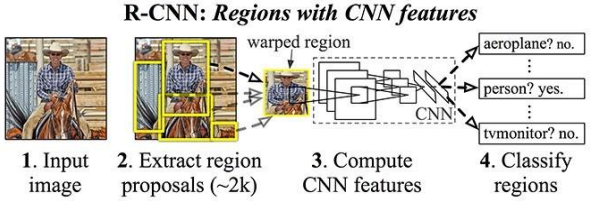
Lapisan input adalah jaringan neural buatan yang memiliki beberapa simpul yang menginput data ke dalamnya. Simpul ini membentuk lapisan input sistem. Lapisan input memproses dan meneruskan data ke lapisan lebih jauh di jaringan neural yang merupakan lapisan tersembunyi yang akan memproses informasi pada tingkat yang berbeda, menyesuaikan perilaku saat lapisan tersebut menerima informasi baru. Jaringan *deep learning* memiliki ratusan lapisan tersembunyi yang dapat digunakan untuk menganalisis masalah dari beberapa sudut yang berbeda [8].

D. R-CNN

Deteksi objek terdiri dari dua tugas terpisah yaitu klasifikasi dan lokalisasi. R-CNN adalah singkatan dari *Convolutional Neural Network* berbasis wilayah(*Region*). Konsep utama di balik seri R-CNN adalah proposal wilayah. Proposal wilayah digunakan untuk melokalkan objek dalam gambar [4].

Proses R-CNN sendiri terdiri dari 3 tahap [9];

1. mencari region atau bagian gambar yang mungkin merupakan sebuah objek, dengan metode *region proposal.*
2. setiap region tersebut kemudian dijadikan input untuk CNN sebagai *feature extractors* dari tiap region tersebut.
3. setiap fitur-fitur yang dihasilkan, kemudian menjadi input untuk SVM (yang akan menghasilkan kelas dari region tersebut) dan linear regressor (yang akan menghasilkan *bounding box*).



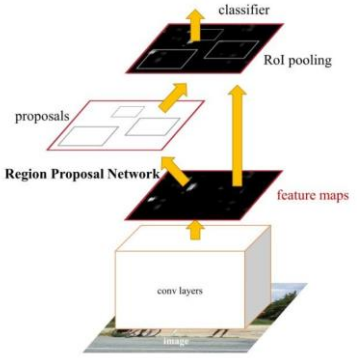
Gambar 4. Arsitektur R-CNN [9]

Seperti yang dapat dilihat pada gambar di atas sebelum mengirimkan gambar melalui jaringan, kita perlu mengekstrak proposal wilayah atau wilayah yang diminati

menggunakan algoritma seperti pencarian selektif. Kemudian, kita perlu mengubah ukuran (membungkus) semua potongan yang diekstraksi dan menyebarkannya melalui jaringan CNN yang nantinya akan memberi klasifikasi[9]

E. Faster R-CNN

Faster R-CNN merupaka algoritma yang menggunakan *Fast R-CNN* dan RPN sebagai arsitektur utamanya. Algoritma ini merupakan pengembangan dari *Fast R-CNN* dengan mengubah bagian selective search pada *Fast R-CNN* menjadi RPN [10].

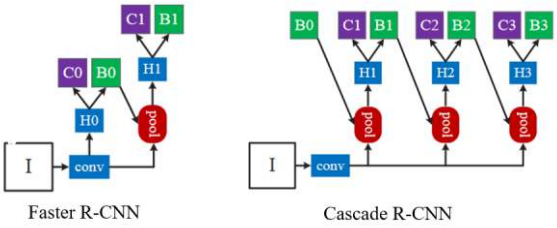


Gambar 5. Arsitektur Faster R-CNN [10]

RPN merupakan proses yang berguna untuk mencari kemungkinan lokasi objek pada gambar yang dimasukkan secara cepat. Lokasi objek yang ada pada gambar memiliki kemungkinan batasan objek dari wilayah yang diketahui yang disebut sebagai *Region of Interest* (RoI). RoI diperkenalkan oleh Ross Girschik tahun 2015 sebagai sebuah pendekatan deteksi objek menggunakan *deep learning*. *Input* yang digunakan pada layer RoI adalah *feature* mAPs yang merupakan *output* dari CNN dengan *multiple convolution layers* dan *max pooling layers*. Sebuah N x N matriks dibuat dengan membagi ruang *feature* mAP menjadi RoI. Kolom pertama merupakan indeks dari citra dan sisa kolom lainnya merupakan koordinat dari RoI, dimulai dari koordinat paling kiri atas hingga kiri bawah.[11] RoI yang sudah ditentukan disebut sebagai *region proposal*. Pada RPN, awalnya citra input diproses dalam jaringan saraf konvolusi untuk menghasilkan *feature* mAP. *Feature* mAP terdiri atas enam bagian, yaitu penentuan *object* dan *non-object* dengan nilai 0–1, koordinat nilai x dan y, serta nilai *weight* dan *height* dari *bounding box*. *Sliding window* ditempatkan pada setiap *feature* mAP dengan ukuran N x N, sesuai dengan setiap *anchor sliding window* yang dibentuk. Setiap *anchor* memiliki titik pusat yang sama, tetapi memiliki aspect *rations* dan *scaling factor* yang berbeda. *Classifier* merupakan proses yang digunakan untuk mengklasifikasikan RoI yang sudah diidentifikasi pada RPN ke dalam kelas atau target yang sesuai. Teknik yang digunakan pada tahap ini adalah CNN [10].

F. Cascade R-CNN

Cascade R-CNN adalah pengembangan dari *Faster R-CNN*. Jika *Faster R-CNN* adalah single-*stage* maka *Cascade R-CNN* merupakan arsitektur *multistage* deteksi objek yang mengatasi masalah penurunan performa dari suatu model *machine learning*. Masalah ini berupa *overfitting* yang terjadi akibat kurangnya dataset. Jika dataset terbatas, maka model bisa saja bekerja dengan baik di dataset tertentu, namun tidak bisa bekerja dengan baik di data set baru [2].



Gambar 6 *Perbandingan* Faster R-CNN *dan* Cascade R-CNN*[2]*

Dari diagram diatas, input gambar yang akan diproses ke conv(*backbone convolutions*) yang merupakan arsitektur yang membentuk jaringan agar bisa mendeteksi gambar, lalu akan di pool yang bertujuan agar ukuran dari matriksnya dikurangi agar jaringan bisa mengenali fitur objek. Setelah itu H1*(network head*) yang merupakan proposal yang tugasnya memprediksi objek akan mengeluarkan input berupa C(*score classification*) dan B(*bounding box*). Hasil C1 dan B2 akan dideteksi lagi dengan langkah yang sama sampai menjadi langkah ketiga yang menghasilkan C3 dan B3[2].

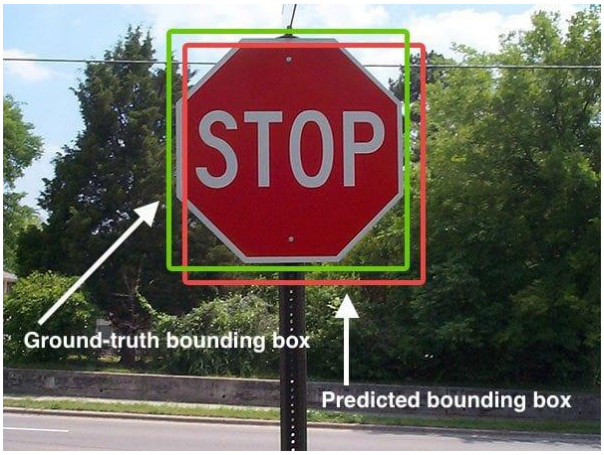
	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv2 [26]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [23]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
RetinaNet [22]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
Faster R-CNN+++ [16]*	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [21]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w FPN+ (ours)	ResNet-101	38.8	61.1	41.9	21.3	41.8	49.8
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Deformable R-FCN [5]*	Aligned-Inception-ResNet	37.5	58.0	40.8	19.4	40.1	52.5
Mask R-CNN [14]	ResNet-101	38.2	60.3	41.7	20.1	41.1	50.2
AttractioNet [10]*	VGG16+Wide ResNet	35.7	53.4	39.3	15.6	38.0	52.7
Cascade R-CNN	ResNet-101	**42.8**	**62.1**	**46.3**	**23.7**	**45.5**	**55.2**

Gambar 7 Perbandingan single-model Cascade R-CNN dengan algoritma lain [2]

Dari gambar diatas bisa dilihat bahwa *Average Precision* pada IoU 0,5 dan 0,75 lebih tinggi dari Algoritma lain. *Average Presision* pada objek *small*, *medium* dan *large* pada *Cascade R-CNN* juga lebih unggul dari algoritma lainnya. Hal ini membuat *Cascade R-CNN* dapat menjadi algoritma yang berkualitas tinggi untuk mendeteksi pelanggaran pada bahu jalan tol[2].

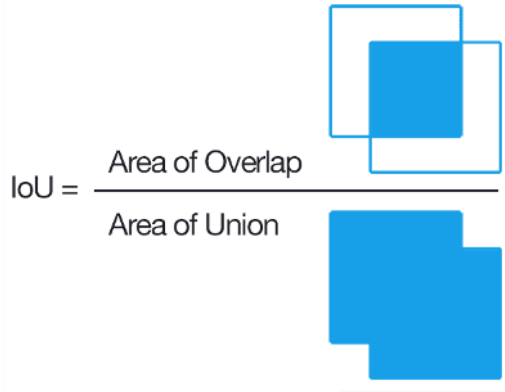
G. Intersection of Union (IoU)

Intersection over Union adalah metrik yang digunakan untuk mengetahui akurasi dari sebuah pendeteksi objek. Agar bisa menggunakan IoU untuk mengevaluasi pendeteksi objek, dibutuhkan *ground-truth bounding boxes* dan predicted *bounding box*es dari algoritma. Jika memiliki kedua hal ini, maka IoU dapat diaplikasikan untuk mengevalusai pendeteksi objek[11].



Gambar 8. Ground-truth bounding box *dan* Predicted bound box[11]

Dari gambar diatas dapat dilihat bahwa pendeteksi objek telah mendeteksi gambar tanda stop. Kotak merah adalah predicted *bounding box* sedangkan kotak berwarna hijau adalah ground-truth *bounding box*. Predicted *bounding box* adalah *bounding box* yang diprediksi oleh algoritma. Sedangkan ground-truth *bounding box* adalah *bounding box* yang berasal dari dataset yang sudah diberi label(contohnya pada gambar diatas missal labelnya adalah "tanda stop"). Label dilakukan secara manual pada dataset. Secara manual IoU dapat dihitung dari kedua *bounding box* ini [11].

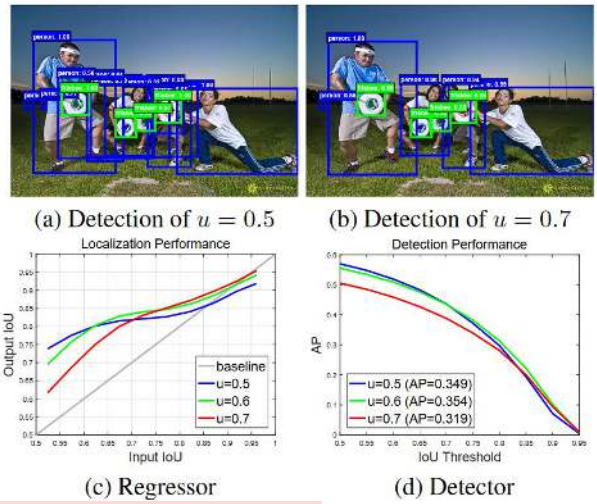


Gambar 9. Perhitungan IoU

Iou adalah hasil pembagian dari *area of overlap* dan *area of union. Area of overlap* adalah irisan dari *ground-truth bounding box* dan *predicted bounding box*. *Area of union* adalah gabungan dari *ground-truth bounding box* dan *predicted bounding box* [11].

Gambar 10 IoU [11]

Rentang IoU adalah 0 sampai 1[11]. Dari gambar diatas semakin sedikit irisan dari prediction *bounding box* dan *ground-truth box* maka IoU semakin tinggi.

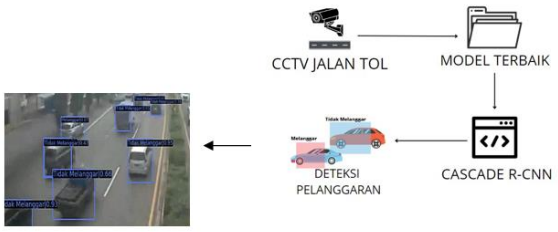


Gambar 11. Hasil AP pada tiap IoU berbeda [2]

Algoritma pendeteksi objek biasanya dilatih dengan IoU 0.5 yang bisa dibilang baik. Dari gambar diatas peforma dari algoritma akan menurun saat IoU meningkat. Hal ini disebabkan oleh *overfitting* ketika *train*ing karena sampel positif akan berkurang secara eksponensial[2].

III. METODE

A. Desain Umum sistem



Gambar 12. Desain umum sistem

Deteksi pelanggaran pada bahu jalan tol menggunakan Cascade R-CNN melalui link CCTV yang diproses pada komputer. Sebelum melakukan deteksi dataset dikumpulkan dengan cara mengambil gambar pada salah satu CCTV jalan tol. Gambar pada setiap dataset diberi label untuk menentukan object kendaraan yang melanggar dan tidak melanggar. Model terbaik yang telah didapatkan digunakan untuk mendeteksi pelanggaran pada CCTV jalan tol. Cascade R-CNN melalui model terbaik memproses gambar untuk menentukan kendaraan yang melanggar dan tidak melanggar.

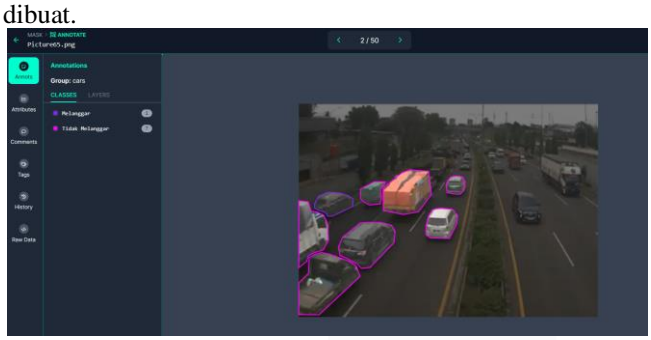
B. Analisis Kebutuhan Dataset

Dataset yang telah digunakan pada penelitian ini sama dengan penelitian yang dilakukan sebelumnya oleh Maulana Heady Yusfian, Casi Setianingsih, dan Ratna Astuti yaitu pada ruas tol JORR-S KM 30+300 dengan ditambah data sebanyak 150 jadi total data sebanyak 250 dalam sati dataset. Dataset diambil dengan cara mengambil tangkapan layar pada CCTV jalan tol, kemudian disimpan dalam bentuk gambar. Gambar yang diambil merupakan gambar yang terdapat kendaraan dengan kondisi melanggar dan tidak melanggar. Dataset kemudian dibagi menjadi data train, data validation, dan data test.

Gambar 13 CCTV jalan tol

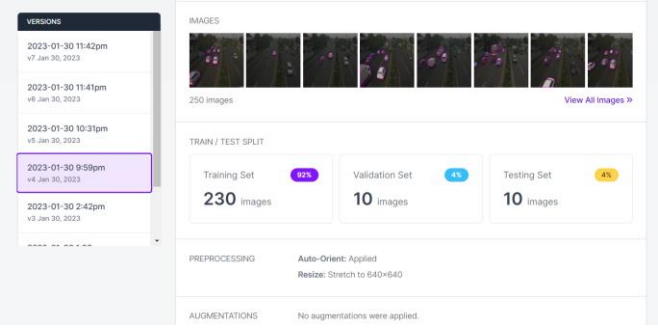
C. Menambahkan label pada dataset dengan Roboflow

Dataset yang telah dikumpulkan di *upload* dan dilabel menggunakan Roboflow, kemudian objek yang akan dideteksi ditandai menggunakan *polygon tools*. Label terbagi menjadi dua kelas yaitu "Melanggar" dan "Tidak Melanggar". Dataset yang telah dilabel diambil kombinasi jumlah data *train, test* dan *valid* dataset di *train* untuk mendapat hasil terbaik dari setiap kombinasi dataset yang dibuat.



Gambar 14. Proses melabel dataset menggunakan roboflow

Dataset yang telah diberi label kemudian dataset dibagi menjadi beberapa versi sesuai dengan kombinasi dataset untuk dianotasi. Dataset dipartisi menjadi 4 kondisi. Kondisi I ialah yang diperlukan yaitu 80% data *train,* 10% data *test*, dan 10% data *validation.* kemudian versi berikutnya *train*ing dataset ditambah jumlahnya hingga menjadi versi yang terakhir yaitu 96% data *train,* 2% data *test*, dan 2% data *validation.* Kemudian dataset diubah dalam format COCO untuk diproses menggunakan *tools mmdetection*.



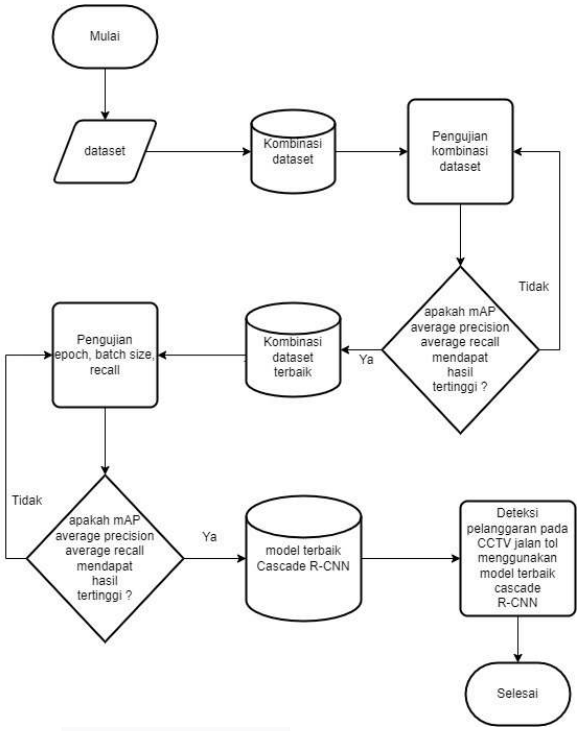
Gambar 15. partisi dataset menggunakan roboflow

dalam format COCO untuk diproses menggunakan tools mmdetection.

Gambar 11
Membuat kombinasi dataset menggunakan roboflow.

D. Proses Pemodelan dan Deteksi Pelanggaran Cascade R-CNN

Gambar 12
Flowchart pembuatan model terbaik.



Gambar 15. *Flowchart* pembuatan model terbaik

Dalam mencari model terbaik kombinasi dataset yang telah dibentuk dilakukan pelatihan. Setelah mendapat hasil terbaik dengan beberapa kombinasi dataset tersebut kemudian dilatih lagi dengan *hyperparameter epoch, batch size,* dan *learning rate*. Dataset yang dilatih pada setiap *hyperparamter,* dilihat parameter mAP, *average precision, dan average recall* untuk menentukan model terbaik. Setelah mendapatkan model terbaik *Cascade R-CNN* salah satu gambar dicoba menggunakan model tersebut untuk melihat hasil dari algoritma *Cascade R-CNN* dalam mendeteksi pelanggaran pada bahu kiri jalan tol. Kemudian model terbaik juga diuji coba pada video jalan tol. Berikut hasil dari mendeteksi pelanggaran pada bahu kiri jalan tol menggunakan model terbaik *Cascade R-CNN*.

E. Hasil Train

Setelah di train model akan menghasilkan banyak hasil metrik seperti dibawah ini.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.762
Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=1000 ] = 0.971
Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=1000 ] = 0.929
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.675
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.775
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.701
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.784
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.784
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.784
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.733
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.798
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.729
```

Gambar 16. Hasil train Cascade R-CNN

COCO membuat 6 metode dalam menghitung AP yaitu pada IoU threshold 0.5-0.95, 0.5, 0,75, pada objek small, medium dan large[12].

```
Average Precision (AP):
  AP                % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50        % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75        % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small          % AP for small objects: area < 32^2
  AP^medium         % AP for medium objects: 32^2 < area < 96^2
  AP^large          % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1          % AR given 1 detection per image
  AR^max=10         % AR given 10 detections per image
  AR^max=100        % AR given 100 detections per image
AR Across Scales:
  AR^small          % AR for small objects: area < 32^2
  AR^medium         % AR for medium objects: 32^2 < area < 96^2
  AR^large          % AR for large objects: area > 96^2
```

Gambar 17. Metrik COCO[12]

Dari gambar diatas banyak sekali metrik yang disediakan coco, namun pada penelitian ini hanya menggunkan empat metrik untuk mengukur performa dari algoritma Cascade R-CNN, yaitu[12]:

1. AP@.5:.95: Ini adalah nilai AP pada Iou 0.5-0.95 dengan steps=0.05(0.5, 0.05, 0.6, 0.65, 0.7, 0.75, 0.85, 0.9, 0.95) jadi totalnya ada 10. Lalu presisi tersebut akan akan dirata-ratakan.
2. AP@.5 : ini adalah AP yang dihitung pada IoU 0.5
3. Precision digunakan untuk menganalisa tingkat keakuratan dari prediksi yang dilakukan.
4. Recall adalah persentase kecenderungan sistem untuk menemukan *True Positive* pada keselutuhan prediksi yang dilakukan.

Pada websitenya [12] COCO mengatakan "AP adalah rata-rata untuk semua kategori. Secara tradisional, ini disebut '*mean Average Precision'* (mAP). Kami tidak membedakan antara AP dan mAP (serta AR dan mAR) dan menganggap perbedaannya jelas dari konteksnya."

F. Hasil Deteksi

Setelah di *train*, *test,* dan *validation* maka algoritma sudah bisa mendeteksi mobil yang melanggar bahu jalan tol pada gambar atau video yang diambil dari CCTV.

Gambar 18. Hasil deteksi pelanggaran menggunakan Cascade R-CNN

Dari gambar diatas algoritma Cascade mendeteksi mobil dengan kelas melanggar atau tidak melanggar pada jalan tol.

G. Kebutuhan Perangkat Keras

Perangkat keras yang digunakan pada penelitian Tugas Akhir ini mempunyai spesifikasi dengan detail sebagai berikut:

1. Processor : Intel Core i7 2.30GHz (4 CPUs), ~2.3GHz
2. RAM : 16 GB
3. Sistem Operasi : Windows 11 Home Single Language 64-bit (10.0, Build 19044)(19041.vb_release.191206- 1406)
4. GPU : Intel(R) HD Graphics 620 dan NVIDIA GeForce MX110, 4GB Display memory

H. Analisis Kebutuhan Perangkat Lunak

Perangkat Lunak yang dibutuhkan untuk pengembangan sistem adalah *Google Colaboratory*, MMdetection, dan Matplotlib. Berikut penjelasan perangkat lunak yang digunakan dalam pengembangan sistem.

1. *Google Colaboratory* adalah software eksekusi kode yang digunakan untuk membuat program Cascade R-CNN.
2. MMDetection digunakan dalam membangun model dan pemrosesan deteksi objek dalam algoritma Cascade R-CNN.
3. Roboflow: Untuk melabel gambar atau dataset

IV. HASIL DAN ANALISIS

A. Skenario Pengujian

Pada tahap pengujian dilakukan uji model dan mengevaluasi model hingga mendapat hasil terbaik dengan konfigurasi hyperparameter berbeda beda. Konfigurasi hyperparameter yang diuji yaitu kombinasi dataset, pengujian *epoch*, pengujian *batch size*, dan pengujian *learning rate*. Model terbaik merupakan model yang mempunyai mAP, precision dan recall yang tinggi.

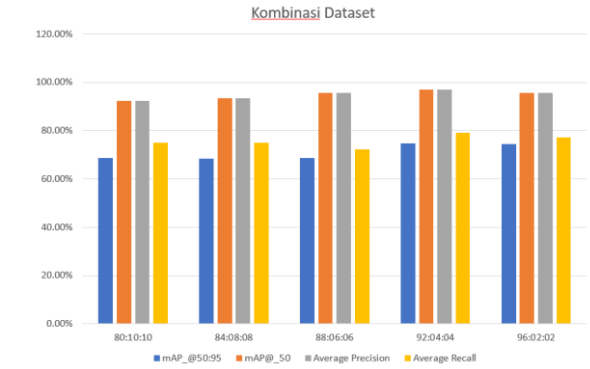
B. Kombinasi Dataset

Dataset yang diuji berjumlah 250 dengan kombinasi dataset yang telah diatur sebagai berikut.

1. 80% data training, 10% data testing, 10% data validation.
2. 84% data training, 8% data testing, 8% data validation.
3. 88% data training, 6% data testing, 6% data validation.
4. 92% data training, 4% data testing, 4% data validation.
5. 96% data training, 2% data testing, 2% data validation.

Tabel 1. Hasil pengujian kombinasi dataset

Partisi Dataset	mAP@.5:.95	mAP@.5	*Average Precision*	*Average Recall*
80:10:10	68,7%	92.3%	92.3%	74,9%
84:8:8	68,5%	93.3%	93.3%	75%
88:6:6	68,8%	95,6%	95,6%	72,3%
92:4:4	74,8%	97,1%	97,1%	79,1%
96:2:2	74,4%	95,7%	95,7%	77,1%



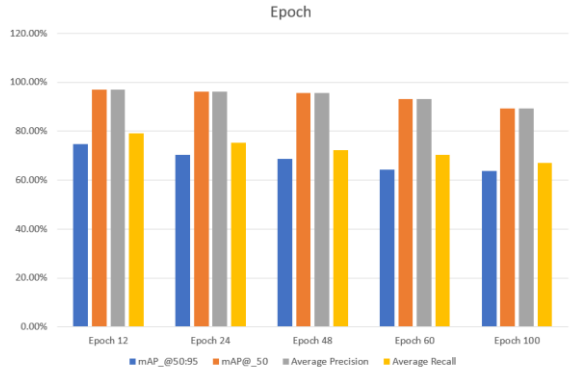
Gambar 19 Grafik hasil pengujian kombinasi dataset

C. Pengujian *Epoch*

Pengujian hyperparameter *epoch* merupakan pelatihan disaat semua data digunakan sekaligus dan didefinisikan sebagai jumlah total iterasi dari semua data pelatihan dalam satu siklus untuk melatih model. Data diuji dengan jumlah epoch 12, 24, 48, 60, 100, 150. Dataset yang digunakan merupakan dataset dengan kombinasi 92% data *training*, 4% data *testing*, 4% data *validation*. Kemudian dataset diuji dengan epoch yang telah ditentukan, berikut hasil pengujiannya.

Tabel 2. Hasil pengujian *epoch*

	mAP_@50:95	mAP@_50	*Average Precision*	*Average Recall*
Epoch 12	74,8%	97,1%	97,1%	79,1%
Epoch 24	70,2%	96.1%	96.1%	75,3%
Epoch 48	68,8%	95,6%	95,6%	72,3%
Epoch 60	64,3%	93,1%	93,1%	70,2%
Epoch 100	63,7%	89,2%	89,2%	67,1%



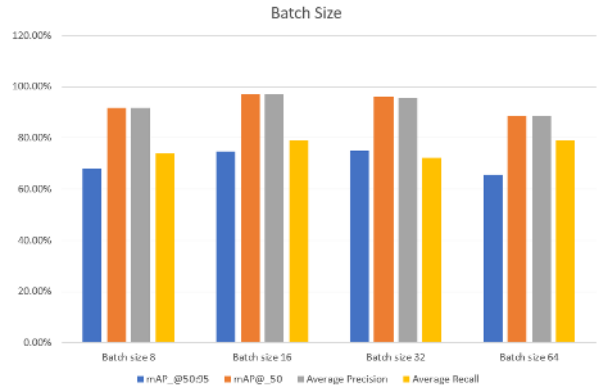
Gambar 10 Grafik pengujian *epoch*

D. Pengujian *Batch Size*

Pengujian hyperparameter *batch size* merupakan hyperparameter yang menentukan jumlah sampel yang diambil untuk mengerjakan model tertentu sebelum memperbarui parameter model internalnya. Batch dapat dianggap sebagai iterasi *for-loop* pada satu atau lebih sampel dan membuat prediksi. Prediksi ini kemudian dibandingkan dengan variabel output yang diharapkan pada akhir batch. *Error* dihitung dengan cara membandingkan keduanya dan kemudian digunakan untuk memperbaiki mode. Data diuji dengan jumlah *epoch* 12 dengan kombinasi dataset 92% data *training*, 4% data *testing*, 4% data *validation*, berikut hasilnya.

Tabel 3. Hasil pengujian batch size

	mAP_@50:95	mAP@_50	*Average Precision*	*Average Recall*
Batch size 8	68,1%	91,7%	91,7%	73,8%
Batch size 16	74,8%	97,1%	97,1%	79,1%
Batch size 32	74,9%	96,1%	95,6%	72,3%
Batch size 64	65,7%	88,5%	88,5%	79,1%
	mAP_@50:95	mAP@_50	*Average Precision*	*Average Recall*



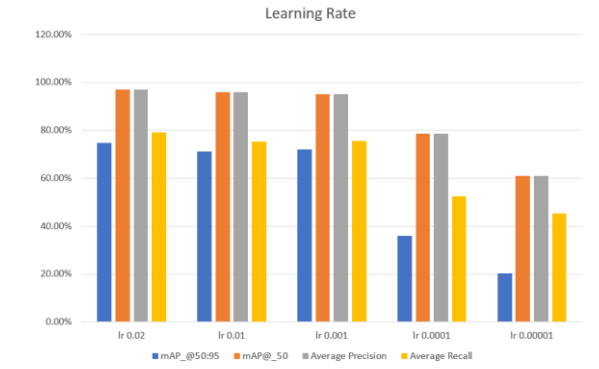
Gambar 21. Grafik pengujian batch size

E. Pengujian *Learning Rate*

Pengujian *learning rate* merupakan hyperparameter yang mengontrol seberapa banyak perubahan model sebagai respons terhadap perkiraan kesalahan setiap kali bobot model diperbarui. Data diuji dengan jumlah *epoch* 12 dengan kombinasi dataset 92% data training, 4% data testing, 4% data validation dan batch size 16, berikut hasilnya.

Tabel 4. Hasil pengujian *learning rate*

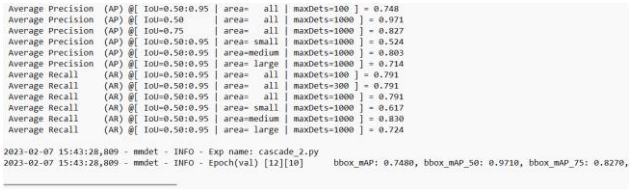
	mAP_@50:95	mAP@_50	*Average Precision*	*Average Recall*
lr 0.02	74,8%	97,1%	97,1%	79,1%
lr 0.01	71,2%	96%	96%	75,4%
lr 0.001	72%	95,1%	95,1%	75,6%
lr 0.0001	35,9%	78,5%	78,5%	52,6%
lr 0.00001	20.2%	61%	61%	45.2%



Gambar 22. Grafik learning rate.

F. Hasil Train Model Terbaik

Model terbaik didapatkan pada partisi dataset 92:4:4, epoch 12, batch size 16 dan learning rate 0,02 dengan AP=97%, AR=79,1%, mAP@.5=97,1% , dan mAP@.5:.95=74,8%

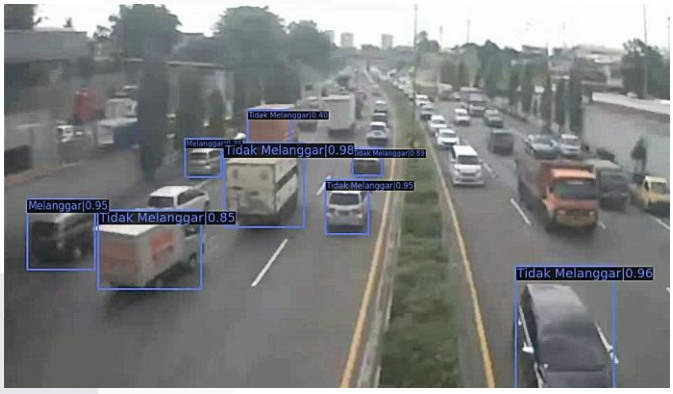


Gambar 23. Hasil *train* model terbaik

G. *Inference* dengan Video

Inference dengan video diperlukan untuk menguji dan memastikan model dapat mendeteksi objek pelanggar pada video maupun pada CCTV secara langsung. Berikut beberapa footage hasil inference dengan video.

Gambar 24 Hasil *inference* video 1



Gambar 25 *Inference* dengan video 2

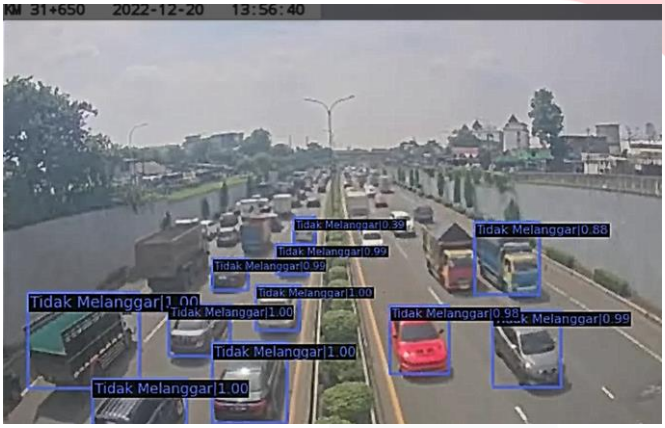
Dapat dilihat dari beberapa *footage* dari video di atas, menunjukkan sistem sudah dapat mendeteksi objek "Melanggar" maupun "Tidak Melanggar" dengan sangat baik. Namun, masih terdapat beberapa kesalahan yang terjadi. Seperti pada beberapa *footage* yang menunjukkan sistem mendeteksi objek "Melanggar" ketika objek di lajur garis pinggir antara bahu jalan tol dan ruas jalan tol. Tidak seperti *inference* dengan *image/gambar,* hasil deteksi objek dengan video tidak menunjukkan adanya *multi labelling* pada objek yang berhasil dideteksi.

H. *Inference* dengan CCTV Ruas Jalan Tol Berbeda

Inference dengan CCTV ruas jalan tol diperlukan untuk menguji dan mememeriksa performa model untuk dapat mendeteksi objek pada CCTV ruas jalan tol yang berbeda

dengan yang ada pada dataset. Untuk penelitian ini, penulis mencoba untuk menguji model dengan dataset yang diambil dari CCTV tol Km 30+200 tol JORR-S dan diuji coba ke CCTV tol Km 31+650. Berikut beberapa *footage* hasil *inference* dengan video.

Gambar 26. *Inference* CCTV berbeda 1



Gambar 27. *Inference* CCTV berbeda 2

Gambar 28. *Inference* CCTV berbeda 3

Gambar 29. *Inference* CCTV berbeda 4

Dari footage yang ditampilkan pada gambar 4.61, 4.62, 4.63 dan 4.64 menunjukkan model dapat mendeteksi 7-10 objek. Model dapat mendeksi semua kelas "Melanggar". Model bisa mendeteksi kelas "Tidak Melanggar" sebanyak 8 dari 10 objek. Maka dari ini model ini bisa mendeteksi objek dengan kelas "Melanggar" dan "Tidak Melanggar" pada CCTV berbeda dengan baik.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan rumusan masalah, tujuan dan analisis pengujian perancangan sistem yang telah dilakukan pada penelitian tugas akhir dapat diambil kesimpulan sebagai berikut.

1. Model terbaik Cascade R-CNN berada pada hyperparameter dengan epoch=12, batch size=16, dan learning rate 0.02
2. Model terbaik Cascade RNN dapat mendeksi objek dengan kelas "Melanggar" dan "Tidak Melanggar" pada CCTV berbeda dengan baik dengan AP=97%, AR=79,1%, mAP@.5=97,1%, dan mAP@.5:.95 =74,8%.

B. Saran

Berdasarkan hasil dari penelitian Tugas Akhir Deteksi Pelanggaran pada Bahu Jalan Tol menggunakan Algoritma Cascade R-CNN, dapat dilakukan pengembangan untuk dihubungkan langsung pada CCTV jalan tol untuk mendeteksi pelanggaran.

REFERENSI

[1] "Jalan Tol," JDIH BPK RI DATABASE PERATURAN, Mar. 15, 2005.

[2] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection," Dec. 2017, [Online]. Available: http://arxiv.org/abs/1712.00726

[3] Y. Amit, P. Felzenszwalb, and R. Girshick, "Object Detection," in Computer Vision, Springer International Publishing, 2020, pp. 1–9. doi: 10.1007/978-3-030-03243-2_660-1.

[4] Y. Tao, Z. Zongyang, Z. Jun, C. Xinghua, and Z. Fuqiang, "Low-altitude small-sized object detection using lightweight feature-enhanced Convolutional Neural Network," Journal of Systems Engineering and Electronics, vol. 32, no. 4, pp. 841–853, Aug. 2021, doi: 10.23919/JSEE.2021.000073.

[5] Institute of Electrical and Electronics Engineers and IEEE Signal Processing Society, 2019 IEEE International Conference on Image Processing (ICIP) : proceedings : September 22-25, 2019, Taipei International Convention Center (TICC), Taipei, Taiwan.
[6] R. Phadnis, J. Mishra, and S. Bendale, "Objects Talk - Object Detection and Pattern Tracking Using TensorFlow," in Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018, Sep. 2018, pp. 1216–1219. doi: 10.1109/ICICCT.2018.8473331.
[7] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.08458
[8] C. Mishra and D. L. Gupta, "Deep Machine Learning and Neural Networks: An Overview," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 6, no. 2, p. 66, Jun. 2017, doi: 10.11591/ijai.v6.i2.pp66-73.
[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Nov. 2013, [Online]. Available: http://arxiv.org/abs/1311.2524
[10] S. Megawan and W. S. Lestari, "Deteksi Spoofing Wajah Menggunakan Faster R-CNN dengan Arsitektur Resnet50 pada Video (Face Spoofing Detection Using Faster R-CNN with Resnet50 Architecture on Video)," 2020. [Online]. Available: https://www.idiap.ch/dataset/replayattack.
[11] Adrian Rosebrock, "Intersection over Union (IoU) for object detection," pyImagesearch, Apr. 30, 2022.
[12] COCO, "COCO Metric," https://cocodataset.org/#detection-eval.