MAKALAH

Sistem Pembantu Deteksi Penggunaan Masker yang Baik dan Benar Menggunakan Object Detection Faster R-CNN

Disusun oleh Tim **Yakuy 2** :

Ardacandra Subiantoro (18/427572/PA/18532)
Arief Pujo Arianto (18/430253/PA/18766)
Chrystian (18/430257/PA/18770)

PROGRAM STUDI S1 ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA

2020

Abstrak.

Daftar Isi

1 Pendahuluan **3**
- 1.1 Latar Belakang . . . 3
- 1.2 Rumusan Masalah . . . 3
- 1.3 Batasan Masalah . . . 3

2 Tujuan dan Manfaat **3**

3 Metode **4**
- 3.1 Dataset . . . 4
- 3.2 Metode Prapemrosesan . . . 4
  - 3.2.1 Konversi ke RGB . . . 4
  - 3.2.2 Normalisasi . . . 4
  - 3.2.3 Konversi ke PyTorch *Tensor* . . . 4
- 3.3 Pembelajaran Model . . . 5
  - 3.3.1 Faster R-CNN . . . 5
- 3.4 Implementasi Weight dari Faster R-CNN Model . . . 9
  - 3.4.1 Perangkat Uji Coba . . . 9
  - 3.4.2 Program WebCam Pendeteksi Masker . . . 9
  - 3.4.3 Program WebCam Pendeteksi Masker dengan Post-processing . . . 10

4 Hasil **12**
- 4.1 Hasil Program . . . 12
  - 4.1.1 Performa Program . . . 12

5 Kesimpulan **13**

6 Lampiran **13**

1 Pendahuluan

1.1 Latar Belakang

Meskipun berbagai penanganan sudah dilakukan tetapi hingga hari ini jumlah kasus COVID-19 terus meningkat. Jumlah kasus total COVID-19 di seluruh dunia sudah mencapai lebih dari 48 juta kasus, dengan lebih dari 1,2 juta kematian akibat COVID-19. Kasus COVID-19 di Indonesia sudah mencapai lebih dari 400 ribu dengan lebih dari 14 ribu kematian. Dampak negatif dari pandemi COVID-19 ini sangat terasa di Indonesia. Direktur Jenderal Pajak Kementerian Keuangan (Kemenkeu) Suryo Utomo membagi dampak pandemi COVID-19 menjadi tiga garis besar [1]. Dampak pertama adalah membuat konsumsi rumah tangga atau daya beli yang merupakan penopang 60 persen terhadap ekonomi jatuh cukup dalam. Hal ini dibuktikan dengan data dari BPS yang mencatatkan bahwa konsumsi rumah tangga turun dari 5,02 persen pada kuartal I 2019 ke 2,84 persen pada kuartal I tahun ini. Dampak kedua yaitu pandemi menimbulkan adanya ketidakpastian yang berkepanjangan sehingga investasi ikut melemah dan berimplikasi pada terhentinya usaha. Dampak ketiga adalah seluruh dunia mengalami pelemahan ekonomi sehingga menyebabkan harga komoditas turun dan ekspor Indonesia ke beberapa negara juga terhenti.

Dengan melemahnya ekonomi di Indonesia ini, banyak perusahaan juga sudah mulai untuk membuka kembali bisnis mereka seperti semula. Berbagai upaya pengawasan pun dilakukan agar setiap orang dapat beraktivitas seperti semula dengan aman tanpa adanya ketakutan tertular COVID-19. Tetapi meskipun berbagai cara pengawasan sudah dilakukan, masih banyak orang yang tidak mematuhi aturan dan tetap beraktivitas seperti biasa. Perlunya banyak pengawasan untuk memperingatkan kembali aturan yang ada di era *new normal* ini adalah salah satu cara agar semua dapat beraktivitas kembali seperti semula serta mengurangi resiko tertular COVID-19.

Tetapi untuk mewujudkan era *new normal* ini dengan baik dan efektif, dibutuhkan pengawasan serta himbauan peraturan era *new normal* ini secara skala besar dan umum sehingga dibutuhkannya banyak sekali tenaga kerja. Namun menggunakan tenaga manusia bukanlah jawaban yang tepat karena tidak efesien serta sangat sulit dan juga menyita banyak waktu apabila tenaga manusia ini mengawasi serta menghimbau setiap orang yang ada.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut permasalahan utama yang akan ditelusuri ialah :

1. Bagaimana penambangan data dapat membantu mewujudkan dan menegakkan aturan pada era *new normal* untuk dapat mengurangi penyebaran COVID-19 yang ada ?

2. Bagaimana cara model dapat mendeteksi serta membedakan pemakaian masker yang baik dan benar ?

1.3 Batasan Masalah

Batasan masalah yang akan kami gunakan adalah sebagai berikut :

- Dataset yang akan digunakan dibatasi pada Kaggle Face Mask Detection milik Larxel[2].

- Jenis gambar yang akan diklasifikasi dibatasi pada memakai masker, tidak memakai masker, dan memakai masker yang tidak benar.

- Metode penambangan data yang akan kami gunakan adalah Real-time Object Detection dengan Arsitektur Faster R-CNN[3].

2 Tujuan dan Manfaat

Tujuan utama, kami mengusulkan menggunakan model Object Detection dengan arsitektur Faster R-CNN untuk membantu mengawasi dan menghimbau penggunaan masker yang benar di tempat umum. Object Detection dengan menggunakan Faster R-CNN ini diharapkan dapat dengan baik

membedakan dan mengklasifikasi penggunaan masker dengan baik dan juga dapat secara cepat melakukan klasifikasi tersebut.

Tujuan kedua, dari hasil penambangan data kami bertujuan untuk langsung menggunakan model tersebut diimplementasikan langsung dengan teknologi serta kesiapa infrastruktur yang ada. Implementasi ini diharapkan dapat membantu tenaga manusia untuk dapat secara otomatis menegakkan serta memperingatkan selalu penggunaan masker yang benar di tempat umum supaya era *new normal* dapat dengan lancar terwujudkan. Manfaat yang kami harapkan adalah peningkatan kesadaran penggunaan masker yang benar dan kemudahan bagi tenaga manusia untuk mengawasi penggunaan masker.

3 Metode

3.1 Dataset

Dataset yang akan digunakan untuk melatih model adalah dataset Kaggle Face Mask Detection milik Larxel[2]. Dataset berisi 853 gambar yang digolongkan menjadi 3 kelas : menggunakan masker, tidak menggunakan masker, dan masker tidak dipakai dengan benar. Tiap gambar disertai dengan keterangan *bounding box* dalam format PASCAL VOC yang menunjukkan lokasi wajah dan masker.

Figur 1: Sample Gambar Pada Dataset Kaggle Face Mask Detection

3.2 Metode Prapemrosesan

Prapemrosesan adalah semua transformasi yang diaplikasikan ke data mentah sebagai bentuk persiapan untuk dimasukkan ke algoritma pembelajaran mesin. Prapemrosesan citra mungkin memiliki efek positif pada kualitas ekstraksi fitur dan hasil analisis gambar [4]. Pelatihan pada algoritma pembelajaran dengan citra mentah tanpa adanya prapemrosesan akan menyebabkan kinerja klasifikasi yang buruk [5].

3.2.1 Konversi ke RGB

RGB (*Red, Green, Blue*) adalah model warna aditif dimana warna merah, hijau, dan biru ditambahkan bersama dengan berbagai cara untuk mereproduksikan sejumlah besar warna [6]. Tujuan utama model warna RGB adalah untuk mendeteksi, merepresentasikan, dan menampilkan gambar-gambar di sistem-sistem elektronik.

3.2.2 Normalisasi

Normalisasi adalah proses yang mengubah kisaran intensitas pixel dan memastikan setiap parameter *input* memiliki ditribusi yang serupa. Tujuan dari normalisasi pada gambar adalah untuk membuat gambar lebih akrab atau terlihat normal bagi indra manusia. Dalam kasus ini, normalisasi dilakukan dengan membagi semua intensitas pixel dalam frame dengan nilai maksimumnya yaitu 255.0.

3.2.3 Konversi ke PyTorch *Tensor*

PyTorch *Tensor* adalah matriks berdimensi banyak yang mengandung elemen-elemen dengan tipe data yang sama. PyTorch *Tensor* memiliki banyak kemiripan dengan numpy array; keduanya

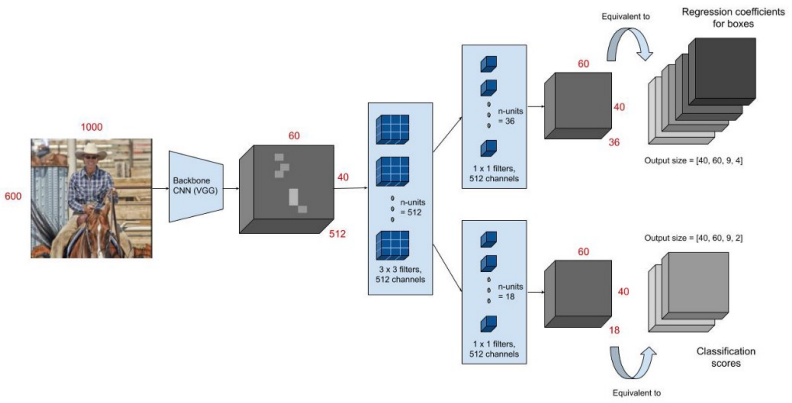
adalah array n-dimensional generik untuk digunakan untuk komputasi numerik. Perbedaan utamanya adalah PyTorch *Tensor* dapat dijalankan pada CPU atau GPU.

Model yang digunakan pada makalah ini dijalankan menggunakan GPU, sehingga mengharuskan gambar masukan masuk dalam bentuk PyTorch *Tensor*. Gambar masukan yang masih dalam model warna RGB dikonversi menjadi PyTorch *Tensor* dengan fungsi torchvision.transforms.ToTensor().

3.3 Pembelajaran Model

3.3.1 Faster R-CNN

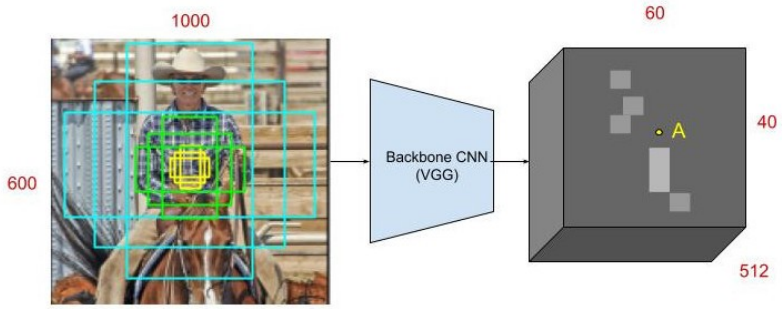
Faster R-CNN pertama kali dipublikasikan pada tahun 2015. Jaringan pendeteksi objek sebelum Faster R-CNN menggunakan algoritma proposal wilayah untuk membuat hipotesis lokasi objek. Jaringan seperti R-CNN dan Fast R-CNN menggunakan algoritma berbasis CPU seperti Algoritma *Selective Search* yang membutuhkan waktu sekitar 2 detik untuk setiap gambar. Jaringan Faster R-CNN memperbaiki masalah ini dengan menggunakan convolutional network untuk menghasilkan proposal wilayah. Perubahan ini mengurangi waktu penghitungan proposal wilayah dari 2 detik menjadi 10 militdetik dan meningkatkan representasi fitur. Faster R-CNN terdiri dari *detection pipeline* yang menggunakan RPN sebagai algoritma proposal wilayah dan Fast R-CNN sebagai jaringan detektor.[7]



Figur 2: Arsitektur RPN

RPN (*Region Proposal Network*) dimulai dengan memasukkan gambar input ke dalam Backbone CNN. Gambar input diubah ukurannya sehingga sisi yang pendek 600px dan sisi yang panjang tidak lebih dari 1000px. Output dari Backbone CNN biasanya jauh lebih kecil dari ukuran gambar input, tergantung parameter stride dari Backbone CNN. Contoh Backbone CNN yang dapat digunakan adalah jaringan VGG atau ZF-Net dengan stride 16. Hal ini berarti pixel-pixel yang bersebelahan pada output jaringan sama dengan dua pixel yang terpisah sejauh 16 pixel pada gambar input.

Untuk setiap titik pada *output feature map*, jaringan harus belajar apakah objek terdeteksi pada gambar input pada lokasi tersebut dan mengestimasi ukurannya. Hal ini dilakukan dengan meletakkan *anchors* pada gambar input untuk setiap lokasi pada *output feature map*. *Anchors* mengindikasikan objek-objek yang mungkin terdapat pada lokasi tersebut dengan beragam ukuran dan *aspect ratio*.



Figur 3: Contoh ”Anchors” pada Gambar Input dari Lokasi Titik A pada *Feature Map*

Jaringan bergerak melewati setiap pixel pada *output feature map* dan memeriksa apakah *anchors* yang mencakupi sebuah area pada gambar input mengandung objek atau tidak, lalu mengoptimalkan koordinat-koordinat *anchors* untuk menghasilkan kotak-kotak pembatas yang disebut proposal wilayah.

Konvolusi 3x3 dengan 512 unit diaplikasikan ke *feature map* untuk menghasilkan 512-d *feature map* untuk setiap lokasi. Lalu diikuti oleh 1x1 convolution layer dengan 18 unit untuk klasifikasi objek, dan 1x1 convolution dengan 36 unit untuk regresi kotak pembatas. 18 unit dari cabang klasifikasi akan mengeluarkan output dengan ukuran (H, W, 18) yang memberikan probabilitas apakah setiap titik dalam *backbone feature map* mengandung objek atau tidak di dalam semua 9 *anchors* pada titik. 36 unit dari cabang regresi akan mengeluarkan output dengan ukuran (H, W, 36) yang memberikan 4 koefisien regresi untuk setiap 9 *anchors* untuk setiap titik dalam *backbone feature map*. Koefisien regresi meningkatkan akurasi koordinat *anchors* yang mengandung objek.

Output feature map terdiri dari 40x60 lokasi, dengan total *anchors* sekitar 20 ribu. Ketika melatih model, semua *anchors* yang melewati batas gambar dihiraukan supaya tidak berkontribusi dalam penghitungan loss. Sebuah *anchor* dianggap sampel positif bila memenuhi salah satu kondisi :

1. *Anchor* memiliki IoU (*Intersection over Union*) tertinggi dengan kotak *groundtruth*.
2. *Anchor* memiliki IoU lebih besar dari 0.7 untuk setiap kotak *groundtruth*.

Sebuah *anchor* dianggap sampel negatif bila IoU dengan semua kotak *groundtruth* kurang dari 0.3. *Anchor* yang tidak termasuk positif dan negatif akan dihiraukan saja.

Setiap *mini-batch* untuk melatih RPN berasal dari sebuah gambar. 128 sampel positif dan 128 sampel negatif dipilih secara acak untuk menghasilkan batch. *Training loss* dari RPN adalah *multi-task loss* yang dihitung sebagai berikut :

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

i adalah index dari *anchor* dalam *mini-batch*. *Classification loss* adalah *log loss* dari dua kelas (objek vs bukan objek). p_i adalah nilai output dari cabang klasikasi untuk *anchor* i, dan p_i^* adalah label *groundtruth*.

Regression loss diaktivasi hanya jika *anchor* mengandung objek. t_i adalah output prediksi dari cabang regresi dan terdiri dari 4 variabel $[t_x, t_y, t_w, t_h]$. *Regression target* dihitung dengan rumus berikut :

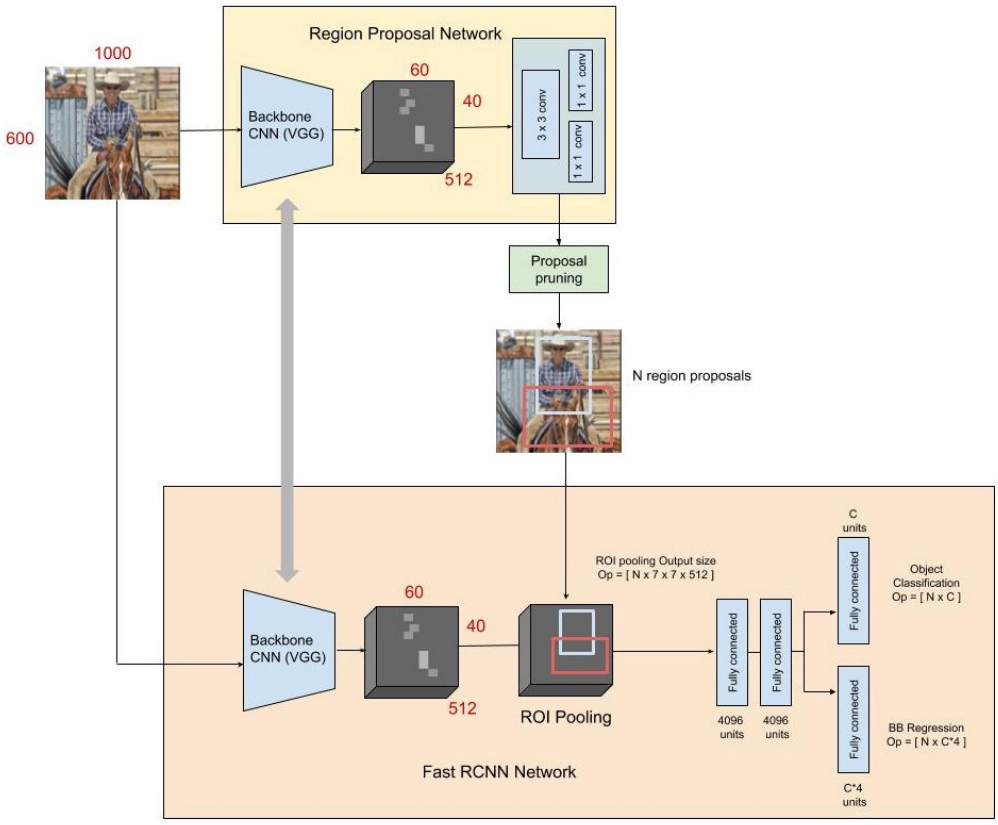
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a)$$

x, y, w, h adalah koordinat (x, y) dari titik tengah kotak pembatas dan tinggi(h) dan lebar(w) kotak. x^*, y^*, w^*, h^* adalah koordinat dan dimensi kotak-kotak *groundtruth*. Setiap *anchor* memiliki

regresor yang berbeda, sehingga weight nya berbeda-beda. Ketika waktu pengujian, output regresi t_i bisa diaplikasikan ke kotak *anchor*, dan parameter x, y, w, h untuk kotak pembatas prediksi bisa dihitung dari rumus :

$$t_{\mathrm{x}} = (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \quad t_{\mathrm{w}} = \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}})$$

Pada waktu pengujian, *anchors* dari setiap gambar melewati beberapa langkah *post-processing* untuk mengirimkan kotak-kotak pembatas objek.Koefisien regresi diaplikasikan ke *anchor* untuk lokalisasi yang tepat. Untuk kumpulan kotak-kota yang tumpang tindih, kotak yang memiliki IoU lebih dari 0.7 dengan kotak lain akan dibuang. Hanya proposal kotak pembatas sejumlah *n* terbaik dari RPN akan dipilih.



Figur 4: Arsitektur Faster R-CNN Lengkap (RPN + Fast R-CNN)

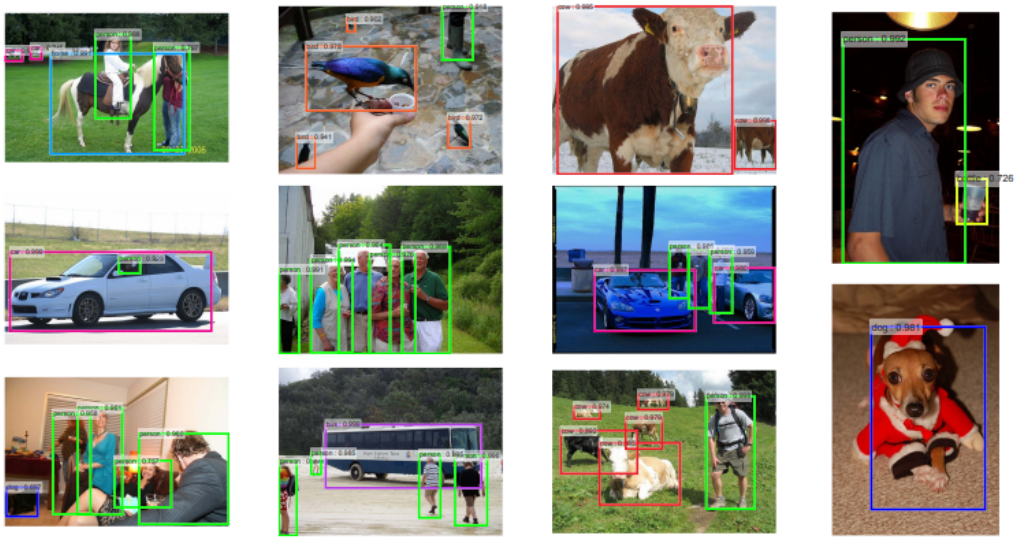
Arsitektur Faster R-CNN lengkap terdiri dari RPN sebagai algoritme proposal wilayah dan Fast R-CNN sebagai jaringan pendeteksi. Jaringan pendeteksi Fast R-CNN juga terdiri dari Backbone CNN, ROI pooling layer, dan fully-connected layer diikuti dengan dua cabang untuk klasifikasi dan regresi kotak pembatas. Gambar input dimasukkan ke Backbone CNN untuk mendapatkan *feature map*. Salah satu keuntungan menggunakan RPN sebagai penghasil proposal adalah pembagian *weights* antara backbone RPN dan backbone Fast R-CNN. Lalu, ROI pooling layer melakukan langkah-langkah di bawah :

1. Mengambil daerah proposal dari *feature map* backbone.
2. Membagi wilayah menjadi sub-wilayah dengan jumlah tertentu.
3. Melakukan *max-pooling* dari sub-wilayah untuk mendapatkan output dengan ukuran tertentu.

Output ROI pooling layer dimasukkan ke dua fully-connected layer, lalu fitur-fitur dimasukkan ke cabang klasifikasi dan regresi. Cabang klasifikasi dan regresi berbeda dengan cabang klasifikasi dan regresi pada RPN. Disini classification layer memiliki C unit untuk setiap kelas pada tugas deteksi. Fitur-fitur masuk melewati softmax layer untuk mendapatkan nilai klasifikasi, probabilitas proposal masuk ke dalam suatu kelas. Koefisien regresi digunakan untuk meningkatkan akurasi dari kotak pembatas. Setiap kelas memiliki regresor masing-masing dengan 4 parameter sesuai dengan C*4 unit output pada regression layer.

Untuk memaksa jaringan untuk membagi *weight* Backbone CNN antara RPN dan Fast R-CNN, digunakan proses latihan 4 langkah :

1. RPN dilatih secara independen seperti penjelasan di atas.
2. Jaringan pendeteksi Fast R-CNN juga dilatih secara independen.
3. *Weight* RPN diinisialisasi dengan *weight* Faster R-CNN, lalu dilatih untuk tugas proposal wilayah. *Weight* pada lapisan-lapisan yang sama antara RPN dan Fast R-CNN tetap sama, hanya lapisan yang ada di RPN saja yang diubah.
4. Dengan menggunakan RPN baru, jaringan pendeteksi Fast R-CNN juga dilatih lagi. Hanya lapisan-lapisan yang ada di Fast R-CNN saja yang berubah *weight* nya.



Figur 5: Hasil Faster R-CNN dengan Backbone VGG

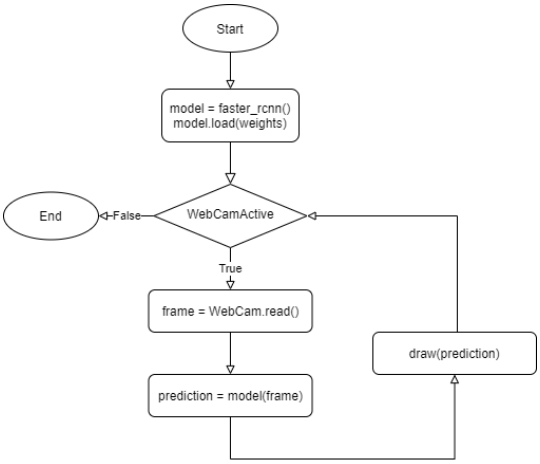
3.4 Implementasi Weight dari Faster R-CNN Model

3.4.1 Perangkat Uji Coba

Hasil output pembelajaran Faster R-CNN adalah *weights* yang dapat digunakan secara langsung untuk memuat kembali model dan memprediksi data baru dengan cepat. Untuk mengimplementasikan model secara langsung atau *realtime* kami menggunakan perangkat keras : CPU Intel(R) i5-3470, GPU Nvidia GTX 960 2GB. Perangkat lunak dan *libraries* : Python 3.7.7, mutils, OpenCV, pytorch, torchvision, numpy, pandas, dan ManyCam.

Dalam percobaan ini kami membuat 2 model program yang memiliki keunggulan masing-masing.

3.4.2 Program WebCam Pendeteksi Masker



Figur 6: Program WebCam Pendeteksi Masker Flowchart

Tugas dari program ini secara umum dapat dibagi menjadi 2 bagian :

1. Proses Memuat *Weights*

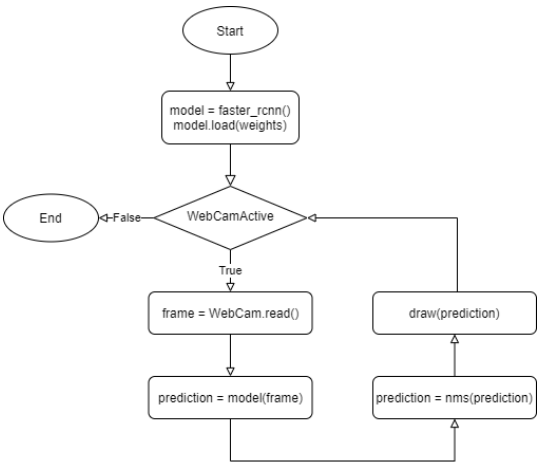
Sebelum program dapat melakukan prediksi, model harus di inisialisasi atau dibuat terlebih dahulu dan selanjutnya model akan dimuat dengan *Weights* yang ada dari hasil proses pembelajaran *transfer learning* Faster R-CNN.

2. Looping ketika input WebCam Aktif

Setelah model sudah siap untuk melakukan prediksi, akan dijalankan selama WebCam aktif maka program akan membaca input yang berasal dari WebCam tersebut dan menyampaikan input tersebut kepada model. Model akan melakukan *Forward Pass* menghasilkan prediksi, informasi dari prediksi ini (labels, scores, boxes) akan digambarkan pada layar, setelah proses ini selesai maka program akan mengulang mengambil input baru sampai program di tutup oleh pengguna.

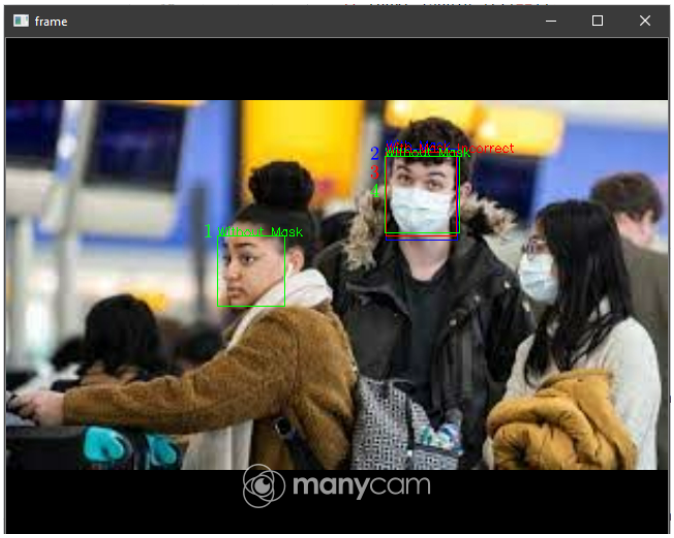
Keunggulan dari program ini adalah kesederhanaannya, program ini mengandung seluruh elemen dasar yang dibutuhkan untuk membuat program deteksi Faster-RCNN secara *realtime*.

3.4.3 Program WebCam Pendeteksi Masker dengan Post-processing



Figur 7: Program WebCam Pendeteksi Masker dengan Post-processing Flowchart

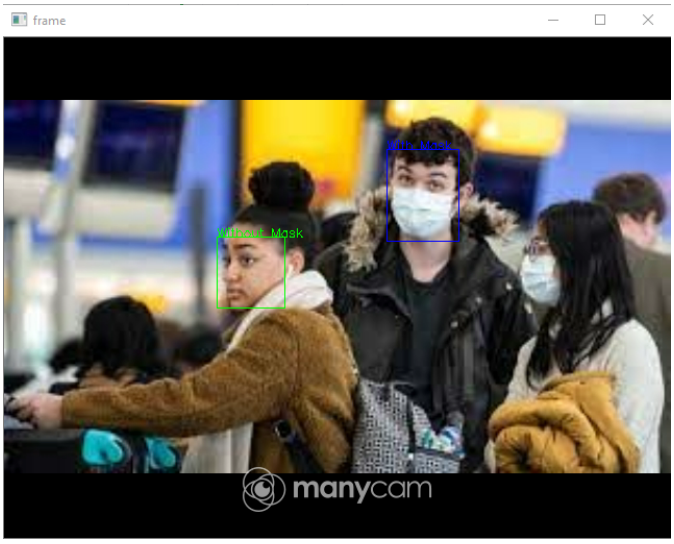
Kami mengusulkan program menggunakan post-processing *non-maximum suppression* untuk memperjelas prediksi yang memiliki *overlapping bounding box* dengan mengeliminasikan atau tidak memakai prediksi yang memiliki skor rendah dalam kotak yang sama. Hal ini terjadi karena perbedaan orang ”memakai masker”, ”tidak memakai masker”, dan ”tidak memakai masker dengan baik” memiliki beberapa fitur yang sama seperti memiliki mata, bentuk wajah hal ini menyebabkan label terdeteksi tetapi memiliki skor yang rendah dan sebaiknya tidak ditampilkan.



Figur 8: Hasil prediksi program.

Hasil Prediksi

i	Label	Score
1	Without Mask	**0.8836**
2	With Mask	**0.8032**
3	With Mask Incorrect	0.2569
4	Without Mask	0.2145



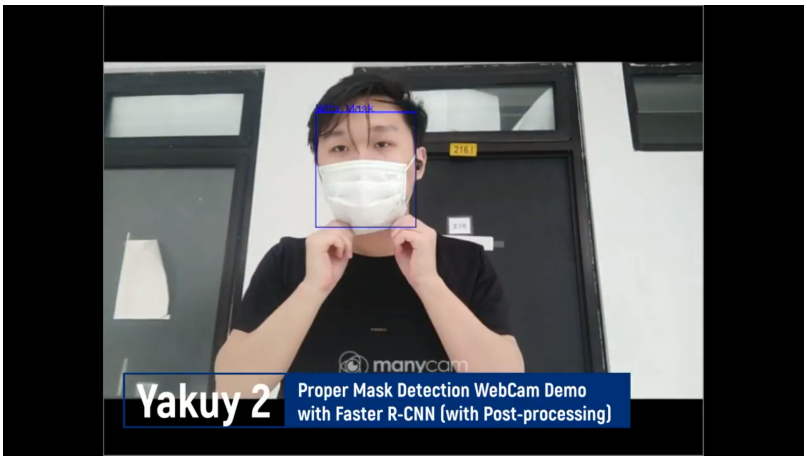
Figur 9: Hasil prediksi program dengan non-maximum suppression.

Hasil Prediksi Menggunakan Post-processing

i	Label	Score
1	Without Mask	**0.8836**
2	With Mask	**0.8032**

4 Hasil

4.1 Hasil Program



Figur 10: Hasil Video Demo dapat dilihat selengkapnya di youtu.be/dnhFLPc0pXI

Hasil dari pembelajaran model Faster R-CNN sudah sangat baik dapat membedakan orang yang menggunakan masker dengan benar atau tidak memakai masker. Hasil model juga dapat mengeneralisasi dengan baik, berdasarkan dari hasil program model masih dapat melakukan *object detection* pengguna masker yang belum pernah dilihat.

4.1.1 Performa Program

```
C:\Users\Chrystian\Desktop\Project\Codig-3.0\demo>python -W ignore webcam.py
GPU CUDA : True
LOG : Load Weight Successfully
[INFO] elasped time: 61.96
[INFO] approx. Prediction per second: 3.15
```

Figur 11: *screenshot hasil benchmark* dari Program WebCam.

```
C:\Users\Chrystian\Desktop\Project\Codig-3.0\demo>python -W ignore webcam_nms_postprocessing.py
GPU CUDA : True
LOG : Load Weight Successfully
[INFO] elasped time: 59.30
[INFO] approx. Prediction per second: 3.10
```

Figur 12: *screenshot hasil benchmark* dari Program WebCam dengan Post-processing (nms).

Program	Waktu Berlalu	Rata-rata Prediksi/s
WebCam Faster R-CNN	61.96	3.15
WebCam Faster R-CNN + Post-processing (nms)	59.30	3.10

Dengan menggunakan Nvidia GTX 960 2GB dan Intel® Core™ i5-3470 Processor, hasil *benchmark* menunjukan Faster R-CNN dapat dengan cepat memprediksi input walaupun dengan GPU yang relatif lama. Ketika menggunakan Post-processing program harus melakukan komputasi fungsi tambahan membuat rata-rata turun tetapi tidak terlalu signifikan atau sekitar -1.58% dari program awal.

5 Kesimpulan

Ketika akan mengimplementasikan model ini maka infrastruktur atau alat yang dibutuhkan adalah sebuah cctv, (setidaknya) sebuah monitor, dan sebuah komputer. Pada hal ini komputer akan berguna sebagai alat untuk menghubungkan cctv dengan monitor dan menjadi tempat untuk model melakukan prediksi kepada gambar input. Kemudian cctv akan berguna menjadi alat untuk mendapatkan gambar input yang kemudian akan diprediksi oleh model. Cctv yang digunakan akan dihubungkan langsung dengan komputer dan gambar input dari cctv akan diproses frame by frame oleh model (Untuk sementara, tolong yg baca ingetin saya edit ini). Setelah itu monitor juga akan dihubungkan dengan komputer, monitor digunakan untuk menampilkan hasil dari input yang diambil dari cctv yang sudah didapatkan hasil prediksinya (ini juga).

Figur 13: Visualisasi Implementasi

Untuk jumlah monitor sendiri tidak terbatas pada satu monitor saja, jumlah monitor dapat disesuaikan dan ditempatkan menyesuaikan dengan kebutuhan pada masing-masing tempat. Dalam monitor nanti gambar yang ditampilkan berdasarkan hasil prediksi setiap orang yang tertangkap dalam cctv akan diberikan bounding box sesuai dengan kategori yang cocok dengan mereka. (abis itu jelasin warna warna tiap kategori tp karena belom tau jadi ingetin nanti). Lalu jika dalam monitor terdapat orang-orang yang tidak menggunakan masker atau menggunakan masker dengan tidak benar maka pengawas/petugas yang berjaga disekitar dapat menegur orang-orang tersebut sesuai dengan kebijakan pada masing-masing tempat.

6 Lampiran

Referensi

[1] N. Zuraya. *Tiga Dampak Besar Pandemi Covid-19 bagi Ekonomi RI*. URL: https://republika.co.id/berita/qdgt5p383/tiga-dampak-besar-pandemi-covid19-bagi-ekonomi-ri. (2020).

[2] Larxel. *Face Mask Detection*. May 2020. URL: https://www.kaggle.com/andrewmvd/face-mask-detection.

[3] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015, pp. 91–99. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[4] S. Krig. "Image pre-processing". In: *Computer Vision Metrics* (1993), pp. 56–111. DOI: 10.1007/978-3-319-50490-23.

[5] K. K. Pal and K. S. Sudeep. *Preprocessing for Image Classification by Convolutional Neural Networks*, pp. 1778–1781. ISBN: 9781509007745.

[6] Robert Hirsch. *Exploring Colour Photography: A Complete Guide*. ISBN: ISBN 1-85669-420-8.

[7] Shilpa Ananth. *Faster R-CNN for object detection*. Aug. 2019. URL: https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46.