PEREKAMAN OTOMATIS BERDASARKAN DETEKSI OBJEK MANUSIA PADA CCTV MENGGUNAKAN METODE *YOU ONLY LOOK ONCE V3 (*YOLOV3)

Mirwan Abdurrahman Hakim[1*)], Tatang Rohana[1)], Dwi Sulistya Kusumaningrum[1)]
[1)] Teknik Informatika, Universitas Buana Perjuangan Karawang, Karawang
*Email Korespondensi : if16.mirwanhakim@mhs.ubpkarawang.ac.id

ABSTRAK

CCTV selama ini memiliki kekurangan dalam hal penggunaan penyimpan, sementara pengawasan dilakukan penuh selama 1 × 24 jam, dan membuat kamera menulis frame terus menerus. Penyimpanan pun dengan drastis terpakai sehingga cepat penuh. Hal ini pun membuat storage seperti HDD, akan berkerja keras sehingga keawetan storage HDD tidak terjamin lama. Solusi yang bisa dilakukan adalah dengan memanfaatkan pengenalan objek sebagai kondisi. Frame hanya akan ditulis pada penyimpan ketika kamera mendeteksi adanya objek manusia. Metode Object Detection yang digunakan adalah YOLOv3, *dataset* dilatih dan menjadi model latih lalu diterapkan secara realtime. Model yang dilatih pada penelitian ini memiliki akurasi 100%, dan setiap objek yang terdeteksi berhasil menjadi kondisi kamera menulis frame pada penyimpan dengan ukuran video paling besar 1,6mb dengan durasi waktu 18 detik.

Kata kunci: CCTV, *Computer Vision, Object Detection, Scurity System, YOLOv3*

ABSTRACT

In recent years CCTV has shortcomings in terms of storage usage, while full surveillance is 1 x 24 hours, and makes the camera write frames continuously. Storage is drastically used so it fills up quickly. This also makes storage such as HDD, will work hard so that the durability of HDD storage is not guaranteed for long. The solution that is able to be done is to use object recognition as a condition. Frames only wrote to the storage when the camera detects a human object. The Object Detection method used was YOLOv3, the *dataset* was trained and become a *training* model then applied in realtime. The model trained in this study has an accuracy of 100%, and each detected object succeeds in becoming a condition for the camera to write a frame to storage with the largest video size of 1.6MB with a duration of 18 seconds.

Keywords: CCTV, Computer Vision, Object Detection, Scurity System, YOLOv3

PENDAHULUAN

Banyaknya terjadi tindak kejahatan membuat sebagian orang memilih untuk memantau rumah, perkantoran, mall, bahkan parkiran dengan memasang CCTV (Closed Circuit Television) [1]. CCTV dapat memantau peristiwa yang terjadi tanpa harus dijaga oleh manusia, dengan begitu selama 24 jam penuh tempat yang dipasangi CCTV dapat dipantau terus menerus. Namun yang jadi masalah adalah CCTV yang merekam frame by frame secara terus menerus membuat memori tidak digunakan dengan cara efesien, karena akan merekam kejadian yang bahkan tidak penting. Hal ini mengakibatkan terjadinya penumpukan fail yang tidak berarti. Perkembangan teknologi saat ini di bidang Computer Vision dengan dikembangkannya teknik jaringan syaraf tiruan dimana tujuannya adalah menduplikasi jaringan syaraf manusia. Jaringan syaraf tiruan memungkinkan komputer memiliki kemampuan mengenali objek, dimana objek ini akan dilatih menggunakan algoritma-algoritma yang sudah banyak dikembangkan.

Pada penelitian sebelumnya metode YOLOv3 digunakan sebagai pemantauan arus lalu lintas, dengan *dataset* yang diperoleh dari rekaman CCTV disembilan simpang yang

ISSN Cetak : 2622-1276
ISSN Online : 2622-1284

berbeda, dengan mean average precision (mAP) 97% pada CCTVFIX dan 99% pada CCTV PTZ [2]. Perbandingan Algoritma juga dilakukan pada penelitian [3] antara Faster R-CNN dan YOLOv3 dengan *dataset* UAV, dengan hasil keduanya memiliki kemampuan klasifikasi yang sama namun YOLOv3 memiliki sensitivitas lebih baik dengan akurasi 99,07%. Pada penelitian [4] mendeteksi citra mobil untuk mengetahui jumlah tempat parkir, dengan *dataset* kumpulan citra kendaraan mobil dengan hasil yang menunjukan perbandingan antara GPU dan CPU. GPU dapat mempercepat waktu komputasi dengan rata-rata sebesar 0,179 detik dibandingkan dengan CPU dan menghasilkan akurasi sebesar 100%. Penerapan peningkatan pada multi-scale pada YOLOv3 dengan *dataset* PASCAL, VOC, dan KITTI. Hasilnya dapat meningkatkan mAP sebesar 7,47% pada *dataset* PASCAL, VOC dan 1,77% pada *dataset* KITTI. Jaringan yang ditingkatkan ini memiliki perbandingan ukuran yang sedikit berbeda namun FPS masih lebih unggul saat sebelum ditingkatkan [5]. Pada penelitian [6] dilakukan untuk sistem pengawasan video dengan pendekatan deep learning metode yang digunakan YOLOv3 dengan kumpulan *dataset* dari hasil perekaman video yang menghasilkan akurasi keseluruhan sebesar 93% pada jenis kendaraan. Akurasi terendah adalah sepeda dengan akurasi 30%, mobil 54% dan bus 40%.

Berdasarkan rujukan yang ada penelitian ini akan menggunakan metode YOLOv3, dan menggunakan *dataset* dari google image/Openimage untuk dilatih. Sebetulnya YOLOv3 sudah memiliki model train yang memiliki 80 classes seperti car, cell phone, book, dan sebagainya termasuk person dengan *dataset* MS-COCO. Namun penelitian ini hanya membutuhkan objek manusia untuk dideteksi sehingga akan dilakukan pelatihan menggunakan 1 classes saja yaitu person sebagai objeknya. Sehingga kamera hanya akan mengenali objek manusia sebagai acuannya menerima perintah untuk merekam.

METODE PENELITIAN

Analisa dan Pengumpulan Data

Dataset yang dipakai pada penilitian ini adalah *dataset OpenImage* dengan mengambil 1 kelas saja yaitu '*person*'. *Dataset* ini sudah memiliki *anotasi/label* sehingga tidak diperlukan lagi aktifitas melabeli gambar secara manual menggunakan *tools* yang tersedia. Gambar yang digunakan sebanyak 500 gambar. Selain *dataset,* penelitian ini membutuhkan beberapa fail seperti:

a. Yolo.cfg
   Fail cfg ini merupakan fail parameter konfigurasi yolo untuk melakukan *training* dengan jaringan konvolusi.
b. Fail.data
   Fail ini berisikan informasi jumlah kelas yang digunakan serta lokasi penyimpanan fail validasi, test, names, dan lokasi backup untuk menyimpan bobot hasil *training*.
c. Person.names
   Fail ini meyimpan *string label* atau *string* kelas yang digunakan karena hanya 1 kelas yang digunakan maka fail ini hanya berisi *string 'person'*.
d. Bobot konvolusi
   Bobot ini adalah jaringan konvolusi untuk melakukan *transfer learning*.

YOLO (You Only Look Once)

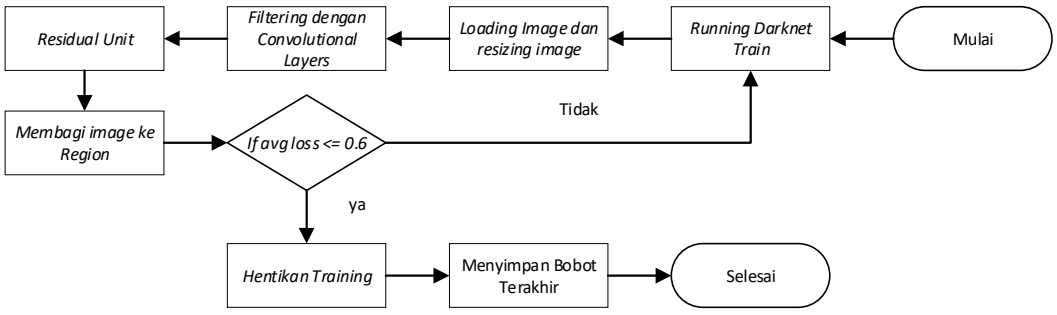
YOLO (*You Only Look Once*) adalah sebuah metode untuk mendeteksi objek untuk pemrosesan secara *realtime*. *YOLO* sendiri mempunyai berbagai macam versi seperti *YOLO*, *YOLOv2* dan yang paling terbaru adalah *YOLOv3* [2]. *YOLO* sendiri diperkenalkan oleh Joseph Redmon pada tahun 2016, dan *YOLO* memiliki pendekatan berbasis wilayah seperti biasanya. Metode ini melewati gambar demi gambar hanya sekali dengan *fully convolutional neural network* (*FCNN*), ini membuatnya cukup cepat dan dapat digunakan secara *realtime*. Pada *YOLOv2* dapat mengatasi masalah kesalahan lokaslisasi yang relatif tinggi dan daya

ingat rendah [7]. *Bounding box prediction, Yolo* pada umumnya melakukan prediksi pada kotak jangkar (*bounding* box) menggunakan *cluster dimension*. Jaringan akan memprediksi 4 buah koordinat untuk setiap kotak pembatas, tx, ty, tw, dan th. Jika sel diimbangi dari kiri atas gambar oleh (cx, cy) dan *bounding box* sebelumnya memiliki lebar dan tinggi pw, ph, maka prediksi sesuai dengan [8]:

$$b_{\times} = \sigma(t_x) + c_x \quad (1)$$
$$b_y = \sigma(t_y) + c_x \quad (2)$$
$$b_w = p_w e^{t_w} \quad (3)$$
$$b_h = p_h e^{t_h`} \quad (4)$$

Pada saat melakukan *training* digunakan jumlah *error loss* kuadrat. Jika *ground truth* untuk beberapa prediksi koordinat adalah $\hat{t}_*$ maka gradient adalah *ground truth* (dihitung dari kotak *ground truth*) dikurangi prediksi: $\hat{t}_* - t_*$. Nilai *ground truth* ini dapat dengan mudah dihitung dengan inverting atau membalik persamaan di atas [8]. *Class Prediction,* dengan menggunakan klasifikasi *multilabel* setiap kotak memprediksi kelas yang bisa dibatasi oleh kotak pembatas. *Softmax* tidak digunakan karena tidak diperlukan untuk hasil kinerja yang baik [8]. Pendekatan dengan *multilabel* memodelkan data dengan lebih baik. Formula ini membantu ketika kita pindah ke domain yang lebih kompleks seperti *Open Images dataset*. Karena pada *dataset* ini ada banyak label yang tumpang tindih, dengan menggunakan *softmax* akan memaksakan asumsi bahwa setiap kotak memiliki satu kelas yang tepat yang sering kali tidak demikian [8]. *Predictions Across Scales, YOLOv3* memprediksi *boxes* pada 3 skala yang berbeda. *YOLOv3* sendiri mengektraksi fitur pada skala tersebut dengan konsep yang mirip dengan fitur jaringan piramida. Pada *dataset COCO* dapat diprediksi 3 kotak pada setiap skala sehingga tensornya adalah $N \times N \times [3 * (4 + 1 + 80)]$ untuk 4 *offset* kotak pembatas, 1 prediksi objek dan 80 prediksi kelas [8]. Selanjutnya akan diambil *map feature* dari 2 *layer* sebelumnya dan *upsample* dengan 2x. *Map feature* dari jaringan sebelumya akan diambil dan menggabungkan dengan fitur *upsampled* dan digabungkan. Hal ini memungkin kita mendapatkan informasi *semantic* yang lebih bermakna dari fitur-fitur *upsampled* dan *finer-grained* informasi dari peta fitur sebelumnya [8]. Terakhir adalah *Feature Extractor,*dimana jaringan YOLO sendiri menggunakan lapisan konvolusional 3 x 3 dan 1 x 1 secara berturut-turut.

Training Dataset



Gambar 1. Alur *Training Dataset*

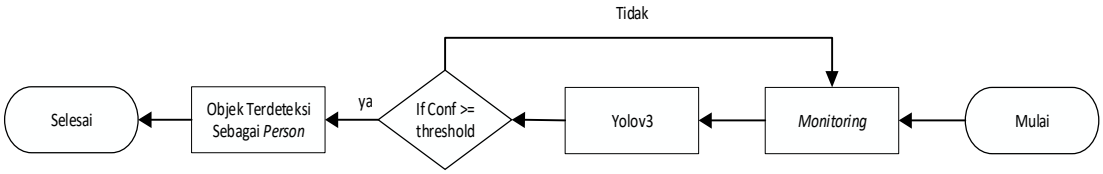
Berdasarkan Gambar 1, blok pertama adalah perintah menjalankan *training* menggunakan *framework* darknet, dengan menggunakan *command ./darknet detector train data/person.data data/Yolov3.cfg data/darknet53.75*. Blok kedua akan dilakukan pemuatan gambar dan akan melakukan *resizing* pada gambar . Seluruh gambar yang sudah ter-*resizing* akan memasuki iterasi pertama dan masuk ke proses *filtering* dengan *Convolutional Layers*. Blok ketiga pada proses *filtering* akan terjadi ekstraksi fitur dengan menggunakan *Convolutional Layers*. Blok keempat setelah memasuki *Convolutional Layers*, maka akan memasuki residual unit dimana akan ada 5 residual blok dan hanya akan diprediksi dalam

ISSN Cetak : 2622-1276
ISSN Online : 2622-1284

3 *downsampled layer* terakhir. Blok kelima akan dilakukan pembagian gambar menjadi wilayah atau *region* lalu memprediksi *bounding box* atau kotak batas dan probabilitas untuk setiap *region*. Hal ini terjadi pada setiap iterasi/*epoch*. Blok keenam bobot atau model akan disimpan pada *folder backup*. Proses ini akan terjadi setiap iterasi mencapai angka 200 dan kelipatannya. Proses diakhiri jika *avg loss* (rata-rata kerugian) mencapai minimum 0.6 atau kurang dari itu.

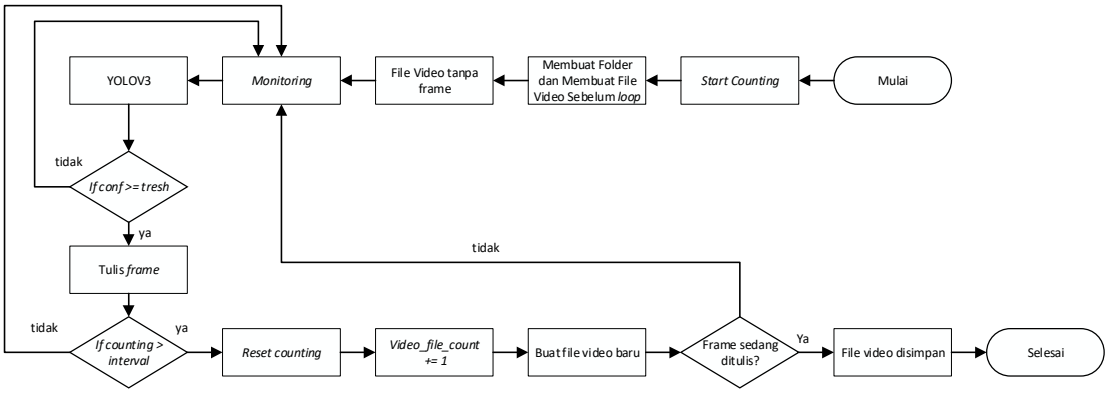
Implementasi Realtime dan Perekaman Otomatis

Hasil *training* dibuat *realtime* untuk mendeteksi keberadaan objek manusia dengan menggunakan *Webcam*. Alur pendeteksian *realtime* seperti Gambar 2:



Gambar 2. Alur Pendeteksian Realtime

Block pertama, kamera akan mememantau dengan menggunakan bahasa python3 dan fungsi dari opencv. Block kedua, Yolov3 merupakan *dataset* yang sudah dilatih dan menjadi model latih. Lalu mengklasfikasikan objek berdasarkan bobot yang sudah dilatih dan terhubung dengan fail *names* yang akan menjadi keluaran *label string*. Model akan dijalankan dan membagi gambar menjadi wilayah dan skala, wilayah yang memilih skor tinggi akan dianggap sebagai objek. Block ketiga, *decision* jika *(confidence)* nilai kepercayaan lebih besar dan sama dengan *(treshold)* ambang batas maka akan dianggap sebagai objek, jika tidak maka akan tetap hanya mememantau. Block keempat, objek terdeteksi saat syarat pada block sebelumnya terpenuhi, lalu menampilkan informasi pada *frame*.



Gambar 3. Alur Realtime dan Perekaman Otomatis

Pada Gambar 3 Blok pertama, akan dimulai *counting* sejak pertama kali program dijalankan. Blok kedua, akan dibuat folder dengan label tanggal dan juga waktu, dan buat fail video kosong pada folder. Blok ketiga, video yang dibuat tidak akan berisikan *frame*. Blok ketiga, kamera akan mememantau dengan menggunakan bahasa python3 dan fungsi dari opencv. Blok keempat, Yolov3 merupakan *dataset* yang sudah dilatih dan menjadi model latih. Lalu mengklasfikasikan objek berdasarkan bobot yang sudah dilatih dan terhubung dengan fail names yang akan menjadi keluaran *label string*. Model akan dijalankan dan membagi gambar menjadi wilayah dan skala, wilayah yang memilih skor tinggi akan dianggap sebagai objek. Blok kelima *decision* jika (*confidence*) nilai kepercayaan lebih besar dan sama dengan (*Treshold*) ambang batas, maka akan dianggap objek dan menampilkan *bounding box*, *label person* dan juga nilai *confidence* dari objek yang dideteksi

pada *frame*. Blok keenam tulis *frame* pada fail video yang telah dibuat sebelumnya, kamera akan merekam video *frame* by *frame* ketika syarat dari blok sebelumnya terpenuhi. Video akan di*encoding* ke dalam bentuk avi. Blok ketujuh, *decision* cek jika *counting* lebih besar dari nilai *interval* waktu yang ditentukan. Jika ya, maka lanjut ke blok delapan. Jika tidak, maka akan tetap mememantau. Blok kedelapan, jika pada blok sebelumnya terpenuhi maka *reset counting*. Blok Sembilan, ketika *counting* di*reset* maka maka *video_fail_count* akan ditambah satu. Blok sepuluh, akan dibuat fail video dengan id baru. Blok sebelas, cek apakah *frame* sedang ditulis. Jika tidak, maka akan tetap mememantau dan jika *frame* ditulis maka simpan video dan selesai.

Pengujian

Pengujian akan dilakukan dengan menguji akurasi model dalam mendeteksi objek manusia secara *realtime*. Berdasarkan penelitian dari [9]. Pengujian dapat dilakukan dengan menentukan jumlah sample uji, lalu mengkalkulasi jumlah data sample yang benar dengan seluruh sample uji. Cara mendapatkan nilai akurasi digunakan rumus:

$$Akurasi = \frac{Jumlah\ Data\ Benar}{Jumlah\ Seluruh\ Data} \times 100\% \quad (5)$$

Selain pengujian model, akan dilakukan pengujian pada perekaman otomatis seberapa besar akurasi kamera mampu melakukan perekaman otomatis disaat ada objek manusia melewati atau berada dalam memantau kamera.

HASIL DAN PEMBAHASAN

Pengumpulan Data

Dataset yang digunakan adalah *dataset* dari *OpenImage*. *Dataset* ini adalah *dataset Opensource* dari *Google*. Seluruh gambar pada *dataset* ini sudah memiliki anotasi. Setiap gambar pada *dataset* akan diambil dengan menjalankan sebuah *source code* yang akan menentukan jumlah kelas dan kelas apa saja yang akan diambil. Dibutuhkan juga sebuah file *class-descriptions-boxable.csv* yang berisikan kelas-kelas yang pada di *OpenImage* dan juga file *train-annotations-bbox.csv* yang berisikan data *annatation*. Seluruh gambar dan label yang sudah terkumpul maka akan dibuat data *person_train.txt* 90% dan juga *person_test.txt* 10%. Didalamnya merupakan sebuah lokasi dari gambar, yang nantinya akan disebar kesetiap *epoch*.

Menentukan Parameter cfg/konfigurasi YOLO

Yolo memiliki berbagai macam model *pre-trainned* dan konfigurasinya. Konfigurasi mengikuti dokumentasi dari [10] dan juga tutorial dari [11] untuk konfigurasi pelatihan *Yolo*, pada file cfg.

Person.data dan Person.names

File *Person.data* ini berisi *classes* serta lokasi dari file *train*, validasi/test, *Person.names* dan juga lokasi *backup* dimana data akan di*backup* dan file bobot hasil perlatihan akan disimpan. *Person.names* merupakan file indeks label *person*. Pada *dataset COCO*, yang dilatih ada 80 *classes* sehingga pada *coco.names* terdapat 80 indeks *label* objek yang berbeda mulai dari *person* hingga *toothbrush*. Karena yang digunakan disini hanya satu kelas saja, yaitu *person* maka dalam file *Person.names* hanya memiliki label *person*.

Persiapan *Training*

Training akan menggunakan *frame*work *Darknet* yang ditulis menggunakan bahasa C. Lalu menjalankan langkah-langkah selanjutnya.

Modifikasi Darknet

Pada dasarnya atau secara default setiap *training* pada *framework darkne*t akan menyimpan *backup* dan bobot/*weight* setiap 100 *epoch* sekali hingga 900 *epoch*. Setelah itu akan

ISSN Cetak : 2622-1276
ISSN Online : 2622-1284

menyimpan bobot pada pada *epoch* 10000 ditunjukan pada Gambar 4. Agar dapat memonitor hasil dan memastikan *loss* (kerugian) berada dinilai 0.6xxx untuk menghentikan proses *training*, maka dilakukan modifikasi, pada file detector.c agar bobot disimpan dengan menyimpan backup setiap 100 *epoch* dan menyimpan bobot setiap 200 *epoch*. Lalu setelah 2000 *epoch* bobot akan disimpan setiap 1000 *epoch* yang ditunjukan pada Gambar 5.

```
134             char buff[256];
135             sprintf(buff, "%s/%s.backup", backup_directory, base);
136             save_weights(net, buff);
137         }
138         if(i%10000==0 || (i < 1000 && i%100 == 0)){
139 #ifdef GPU
140             if(ngpus != 1) sync_nets(nets, ngpus, 0);
141 #endif
```

Gambar 4. Kondisi Nilai Awal *Looping* Penyimpanan Bobot Per Iterasi

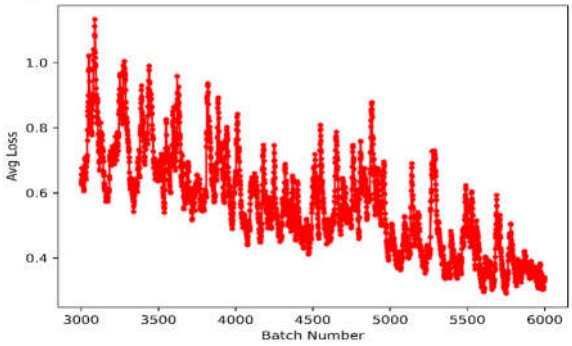
```
134             char buff[256];
135             sprintf(buff, "%s/%s.backup", backup_directory, base);
136             save_weights(net, buff);
137         }
138         if(i%1000==0 || (i < 2000 && i%200 == 0)){
139 #ifdef GPU
140             if(ngpus != 1) sync_nets(nets, ngpus, 0);
```

Gambar 5. Merubah Nilai *Looping* Penyimpanan Bobot Per Iterasi

Cara mengaktifkan pemrosesan menggunakan *GPU* dan *CUDNN* maka modifikasi harus dilakukan juga pada Make File dengan mengubah angka 0 menjadi 1. Lalu dikompail untuk menerapkan perubahan.

Training

Training akan dilakukan menggunakan google colab, maka kita dapat menggunakan *free GPU* yang disediakan oleh google. Kekurangannya, file yang diunggah akan terhapus setelah 12 jam. Sehingga, harus melakukan *training* setiap 12 jam sekali. Menurut [11], semakin kecil nilai *loss* maka semakin lebih baik model yang kita latih. Rekomendasikan menghentikan *training* minimal dengan nilai avg loss 0.6xxxx. *Training* dilakukan hingga mencapai nilai *average loss* yang diinginkan. Hingga penelitian ini menghabiskan 36 jam dengan mendapatkan 6000 *epoch* dapat dilihat pada *batch number* dan nilai *avg loss* berada di 0.333085 ditunjukan pada Gambar 6:



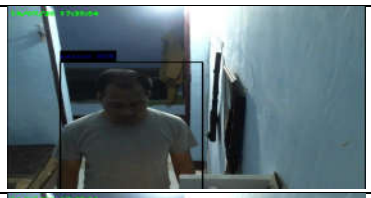
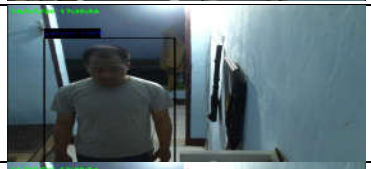
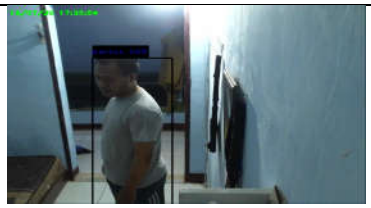
Gambar 6. Gambar Grafik *Avg Loss*

Pengujian Model

Pengujian model dilakukan agar mengetahui akurasi dari model yang telah dilatih, dengan menentukan jumlah sample data agar mendapatkan jumlah data yang benar dari sample

data yang di uji, lalu dikalkulasikan sehingga mendapatkan nilai akurasi. Parameter yang digunakan adalah semua posisi yang memungkinkan dilakukan oleh manusia ketika melakukan aktivitas. Jumlah sample yang diuji adalah 20 sample, setiap sample akan memiliki status apakah terdeteksi atau tidak yang gambarkan kotak pembatasan pada objek manusia. Setiap Objek yang terdeteksi akan memiliki nilai *confidence* atau nilai kepercayaan, ditunjukan pada Tabel 1.

Tabel 1. Tabel Pengujian Model

No.	Gambar	Posisi	Status
1.		Dekat, setengah badan menghadap depan.	Terdeteksi (*confidence*: 95%)
2.		Dekat, setengah badan menghadap depan.	Terdeteksi (*confidence*:100%)
3.		Dekat, setengah badan menghadap samping.	Terdeteksi (*confidence*:99%)
4.		Dekat, setengah badan menghadap samping	Terdeteksi (*confidence*:99%)
5.		Dekat, setengah badan menghadap samping.	Terdeteksi (*confidence*:100%)
. . .			
20.		Dekat, seluruh badan menghadap samping, duduk	Terdeteksi (*confidence*: blp;.100%)
Jumlah terdeteksi:			**20**

Keterangan :
Dekat = Jarak kamera dengan objek manusia sekitar 0.1 meter - 1 meter.
Jauh = Jarak kamera dengan objek manusia sekitar 1,1 meter - 2 meter.

ISSN Cetak : 2622-1276
ISSN Online : 2622-1284

Hasil pengujian dengan sample uji berjumlah 20, dengan seluruh sample yang dapat dideteksi dengan benar. Sehingga, menghasilkan akurasi sebagai berikut:

$$Akurasi = \frac{20}{20} \times 100\% = 100\%$$

Berdasarkan dari bab 3, target yang ingin dicapai adalah lebih atau sama dengan 75%. Jadi model dapat digunakan untuk pengujian selanjutnya yaitu perekaman secara otomatis berdasarkan dari objek yang terdeteksi.

Pengujian Perekaman Otomatis

Pengujian perekaman otomatis adalah tahap akhir, untuk melihat akurasi dari perekaman berdasarkan objek yang terdeteksi. Perekaman hanya memiliki *interval* waktu maksimal 1 menit. Pengujian otomatis dilakukan dengan ukuran gambar yang berbeda-beda. Setiap ukuran gambar dilakukan 5 kali pengujian. Sehingga seluruh pengujian menggunakan 5 ukuran yang berbeda pada *Input Images*. Semakin besar ukurannya maka *frame* yang hasilkan akan semakin sedikit dan mempengaruhi ukuran video. Semakin kecil maka semakin besar *frame* yang dihasilkan, sehingga menghasilkan perbedaan ukuran video walaupun memiliki durasi yang sama yang ditunjukan pada Tabel 2:

Tabel 2. Tabel Pengujian Perekaman Otomatis

No.	Video	Durasi	*Input Images*	Ukuran Video
1.	Video1	31 detik	(640,480)	511,5 kb
2.	Video2	31 detik	(640,480)	385,0 kb
3.	Video3	29 detik	(640,480)	355,9 kb
4.	Video4	24 detik	(640,480)	277,5 kb
5.	Video5	19 detik	(640,480)	229,8 kb
6.	Video1	42 detik	(416,416)	636,1 kb
7.	Video2	18 detik	(416,416)	383 kb
8.	Video3	42 detik	(416,416)	491,1 kb
9.	Video4	18 detik	(416,416)	383 kb
10.	Video5	26 detik	(416,416)	491,1 kb
. . .				
20.	Video5	29 detik	(188, 188)	1,1 mb

Evaluasi

Pengujian model mendapatkan akurasi 100% dengan sampel 20 posisi dan jarak yang berbeda. Model dapat digunakan pada perekaman otomatis, otomatis disini adalah sistem akan menulis *frame* hanya ketika objek terdeteksi, namun tidak akan menulis *frame* ketika objek tidak ada. Objek yang digunakan adalah objek manusia atau *person*.

KESIMPULAN

Berdasarkan hasil penelitian ini, maka dapat disimpulkan bahwa, Akurasi dari model yang telah dilatih memiliki akurasi 100% dalam mendeteksi objek manusia. Dari *sample* yang diuji semua dapat dikenali sebagai objek person. Kemudian hasil penerapan realtime dapat digunakan dan hasil deteksi manusia dengan YOLOv3 menghasilkan akurasi 100%. Sehingga, perekaman otomatis menghasilkan akurasi yang sama.

Agar pengembangan penelitian lebih baik, maka disarankan untuk menambah *Dataset* dua kali lipat untuk mendapatkan *training* yang baik. *Training* dapat dilakukan dengan menggunakan GPU yang memiliki spesifikasi minimal seperti tipe GTX 1080 TI

dengan 11 GB VRAM atau RTX 2070 dengan 6 GB VRAM, supaya *training* dapat dilakukan pada komputer pribadi dan tanpa ada nya jeda seperti pada google colab. Sebaiknya gunakan setting width dan height yang lebih besar pada file cfg dari pada penelitian ini. Namun demikian mungkin akan memakan waktu yang lebih lama dalam pelatihan. Pendeteteksian yang hanya memerlukan objek yang tidak terlalu kecil pada frame, dapat menggunakan YOLOv3 versi tiny. Versi ini menghasilkan frame per second yang lebih tinggi, namun akurasi yang rendah untuk objek jauh atau kecil. Versi ini cocok untuk CCTV pada mesin ATM.

UCAPAN TERIMA KASIH

Naskah ilmiah ini adalah sebagian dari penelitian Tugas Akhir milik Mirwan Abdurrahman Hakim yang dibimbing oleh Tatang Rohana dan Dwi Sulistya Kusumaningrum.

REFERENSI

[1] O. A. Astra and Y. Mardiana, "Rancang Bangun dan Analisa Pengendali CCTV Berbasis Arduino Menggunakan Smartphone Android," *J. Media Infotama*, vol. Vol.14 No., no. 1, pp. 39–50, 2018.

[2] M. Harahap, J. Elfrida, P. Agusman, M. Rafael, R. Abram, and K. Andrianto, "Sistem Cerdas Pemantauan Arus Lalu Lintas Dengan YOLO (You Only Look Once v3)," pp. 367–376, 2019.

[3] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3," *2019 1st Int. Conf. Unmanned Veh. Syst. UVS 2019*, 2019.

[4] S. Jupiyandi *et al.*, "Pengembangan Deteksi Citra Mobil Untuk Mengetahui Jumlah Tempat Parkir Menggunakan Cuda Dan Modified Yolo," vol. 6, no. 4, pp. 413–419, 2019.

[5] M. Ju, H. Luo, Z. Wang, B. Hui, and Z. Chang, "The application of improved YOLO V3 in multi-scale target detection," *Appl. Sci.*, vol. 9, no. 18, 2019.

[6] P. Lestari, D. H. D. Manik, N. L. Br Sihotang, and A. M. Husein, "Video Surveillance System with a Deep Learning Approach," *SinkrOn*, vol. 4, no. 1, p. 263, 2019.

[7] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3," *2019 1st Int. Conf. Unmanned Veh. Syst. UVS 2019*, pp. 1–6, 2019.

[8] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[9] J. Indra, N. Agani, and H. H. Handayani, "Klasifikasi Fertilitas Telur Itik dengan Pengolahan Citra Digital Menggunakan Raspberry Pi," *Techno Xplore J. Ilmu Komput. dan Teknol. Inf.*, vol. 3, no. 2, pp. 68–76, 2018.

[10] Alexey AB, "AlexeyAB/darknet: Windows and Linux version of Darknet Yolo v3 v2 Neural Networks for object detection (Tensor Cores are used)," *GitHub*. 2019.

[11] Sunita Nayak, "Training YOLOv3 : Deep Learning based Custom Object Detector | Learn OpenCV," *Learn OpenCV*, 2019. [Online]. Available: https://www.learnopencv.com/training-yolov3-deep-learning-based-custom-object-detector/.

ISSN Cetak : 2622-1276
ISSN Online : 2622-1284