Program Deteksi Objek menggunakan YoloV5

June 23, 2022

1 Pendahuluan

1.1 Latar Belakang

Computer Vision merupakan suatu teknologi pelatihan mesin dengan memperoleh informasi-informasi melalui citra seperti foto, video, ataupun input visual agar bisa melakukan tujuan tertentu misalnya pendeteksi objek. Pendeteksi objek merupakan kemampuan mesin dalam mendeteksi objek.

Computer vision bisa menggunakan data-data, yang mana data-data tersebut bisa dilakukan training untuk dianalisis agar bisa mengenali suatu objek. YoloV5 merupakan model algoritma generasi kelima untuk mendeteksi objek. Untuk melakukan deteksi objek, kita perlu melakukan tahapan pre-processing data dan processing data. Setelah data dilakukan proses training, data akan ditesting untuk menunjukkan apakah proses training data sudah berhasil.

Di jalan raya, kita sering menemui berbagai objek yang mempengaruhi manusia berkendara. Contohnya adalah petunjuk jalan, zebra cross sebagai tempat penyebrangan pejalan kaki, polisi tidur sebagai pembatas kecepatan, dan hal yang tidak dibuat seperti genangan air pada jalan raya. Hal-hal tersebut tentunya memperngaruhi perjalanan saat kita berkendara. Selain itu, dewasa ini sedang berkembangnya kendaraan berteknologi artificial intelligents yang mana memudahkan pengguna untuk berkendara dengan deteksi objek adalah salah satu fitur di dalamnya. Oleh karena itu, pentingnya teknologi deteksi objek di jalan raya untuk memudahkan pengguna dalam berkendara.

Pada final project matakuliah Pengolahan Citra Video adalah membuat program deteksi objek menggunakanYoloV5

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka dapat ditarik rumusan masalah adalah bagaimana cara mendeteksi objek di jalan raya?

1.3 Tujuan

Berdasarkan rumusan masalah di atas maka tujuan yang akan dicapai yaitu untuk mengetahui cara mendeteksi objek di jalan raya.

2 Program Deteksi Objek

Pada program deteksi objek ini, saya menggunakan model algoritma YoloV5 dengan Python dan proses training dilakukan di Google Colab. Program ini meliputi penginstallan algoritma YoloV5, training, dan testing data.

- Penginstallan YoloV5

```
# install yolov5
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
%pip install -qr requirements.txt

import torch
import utils
display = utils.notebook_init()
```

- Unzip File Training

```
# Unzip
!unzip -q ../fp.zip -d ../
```

- Training Data menggunakan YoloV5

```
# Train YOLOv5s
!python train.py --img 640 --batch 21 --epochs 3
--data data.yaml--weights yolov5s.pt --cache
```

- Testing Data

```
# Testing
!python detect.py --weights runs/train/exp5/weights/best.pt
--img 640 --conf 0.25 --source ../video.mp4
```

3 Dataset yang digunakan

Pada tahapan pre-processing data deteksi objek ini, foto yang tidak bisa dilakukan pendeteksian akan dihapus atau tidak digunakan. Setelah itu dilakukan proses labeling pada foto yang telah diambil. Pada pross labeling menggunakan Roboflow. Roboflow merupakan platform yang digunakan untuk melakukan pre-processing data dan bisa juga melakukan training data sendiri. Namun, pada project ini saya menggunakan Roboflow untuk melakukan labelling data dan melakukan augmentasi pada data, serta membagi data antara data training dan testing.
Data set pada project ini terdiri dari tiga class, yaitu zebracross, polisi tidur, dan genangan. Pada class zebracross terdiri dari sekitar 66 data, class polisi tidur terdiri dari sekitar 147 data, dan class genangan terdiri dari 161 data,

sehingga total data yang dimiliki adalah sekitar 460 data. Namun, di dalam tahapan augmentasi, data akan diperbanyak sesuai dengan pilihan augmentasi yang dilakukan. Pada data pre-processing ini data dibagi menjadi training dan validation. Pada data ini training sebesar 80 persen dan testing sebesar 20 persen.

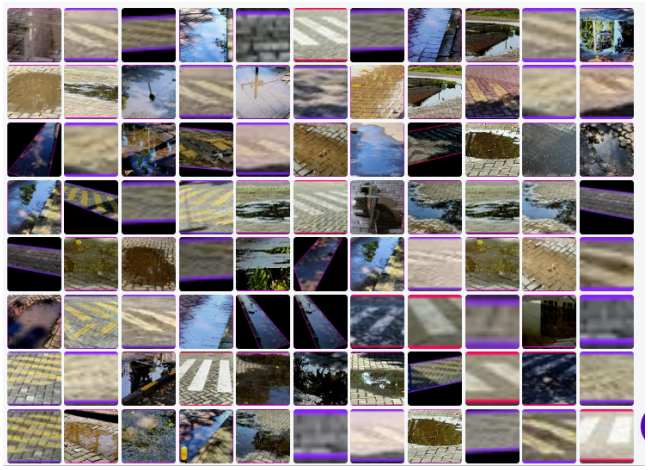


Figure 1: Dataset

Figure 2: Class Zebra Cross

Pada file yaml, class didefinisikan sebagai berikut. Tujuannya untuk mengarahkan program letak data images training dan data validation.

Figure 3: Class Polisi Tidur

Figure 4: Class Genangan

```yaml
train: ../train/images
val: ../valid/images

nc: 4
names: ['-', 'genangan', 'polici', 'zebracross']
```

Figure 5: File yaml

4 Training Data

Setelah proses pre-processing, dilakukan proses training augmentation. Augmentation adalah proses memperbanyak dataset dengan beberapa variasi yang dilakukan seperti flip, rotate, crop, rotation, shear, greyscale, dan lain-lain. Pada project ini saya menggunakan proses augmentation exposure dan brightness agar bisa mendeteksi objek ketika exposurenya tinggi atau terang dan gelap. Pada exposure digunakan -20 persen kebawah karena foto yang terdapat pada data kebanyakan foto yang memiliki brightness yang tinggi. Selanjutnya pada exposure -10 persen dan +10 persen.

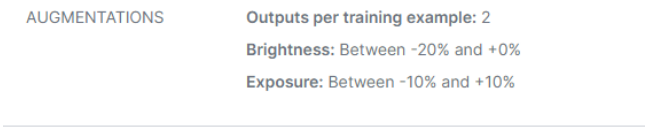
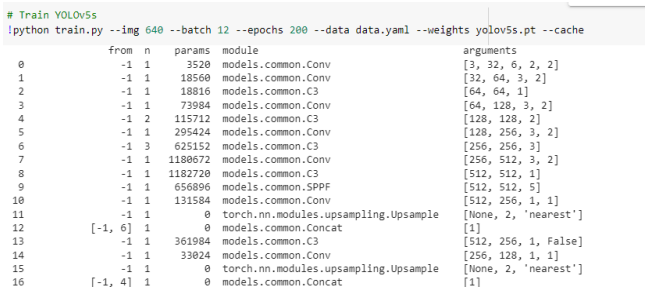


Figure 6: Augmentation

Pada tahapan project mendeteksi objek ini merupakan tahapan dari Machine Learning. Dalam machine learning terdiri dari training data merupakan proses pelatihan suatu mesin untuk mendapatkan model yang sesuai, data testing yaitu menguji coba apakah model atau hasil yang diuji akurat.
Pada proses training program deteksi objek ini menggunakan algoritma YoloV5. Agar objek bisa berhasil dideteksi, kita harus memperkirakan berapa epoch dan batch size yang sesuai agar bisa berhasil melakukan prediksi objek. Pada program ini saya menggunakan epoch sebesar 200 dan batch size sebesar 12.
Epoch merupakan jumlah berapa kali data yang ingin dilakukan training, sedangkan batch size merupakan jumlah data yang sekaligus dimasukkan ke dalam proses training.



```
# Train YOLOv5s
!python train.py --img 640 --batch 12 --epochs 200 --data data.yaml --weights yolov5s.pt --cache
```

	from	n	params	module	arguments
0	-1	1	3520	models.common.Conv	[3, 32, 6, 2, 2]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816	models.common.C3	[64, 64, 1]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	2	115712	models.common.C3	[128, 128, 2]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	3	625152	models.common.C3	[256, 256, 3]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	1182720	models.common.C3	[512, 512, 1]
9	-1	1	656896	models.common.SPPF	[512, 512, 5]
10	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	models.common.Concat	[1]
13	-1	1	361984	models.common.C3	[512, 256, 1, False]
14	-1	1	33024	models.common.Conv	[256, 128, 1, 1]
15	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0	models.common.Concat	[1]

Figure 7: Training Data

5 Deteksi Objek

Setelah melakukan training data, kita bisa melakkan deteksi objek. Data yang sudah ditraining dan ditesting akan disimpan di folder tertentu di dalam folder Yolo. Pada folder tersebut terdiri dari label dan prediksi yang dilakukan oleh algoritma Yolo. Dengan menggunakan size gambar 640, epoch 200 dan batch size sebesar 12, maka akan didapatkan label dan prediksinya sebagai berikut.

Setelah dilakukan prediksi pada proses training tadi, maka kita bisa memasukkan video atau foto untuk dideteksi objeknya. Pada deteksi objek ini saya menggunakan video kemudian diproses pada code berikut.

```
# Testing
!python detect.py --weights runs/train/exp5/weights/best.pt
--img 640 --conf 0.25 --source ../video.mp4
```

Hasilnya adalah sebagai berikut, objek telah berhasil dideteksi Namun, masih

Figure 8: Deteksi objek

terdapat bug error mengenai deteksi objek. Oleh karena itu akan dilakukan riset epoch, batch size, image size, augmentasi, dan review ulang pre-processing data agar bisa mendapatkan deteksi objek yang akurat