Teknik Augmentasi Citra untuk Meningkatkan Kinerja Segmentasi menggunakan Deep Learning

by Erwin Erwin

Submission date: 18-Feb-2022 02:40PM (UTC+0700)
Submission ID: 1765298692
File name: Manual_Book-Erwin-Teknik_Augmentasi.pdf (1.1M)
Word count: 1575
Character count: 9219

Buku Petunjuk Pengguna Aplikasi

(User Manual)

Teknik Augmentasi Citra untuk Meningkatkan
Kinerja Segmentasi menggunakan Deep Learning

OLEH

Dr. ERWIN, S.Si., M.Si
Dr. BAMBANG SUPRIHATIN, S.Si., M.Si

Universitas Sriwijaya
2022

KATA PENGANTAR

Puji syukur Penulis haturkan kehadirat Allah SWT, atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan Buku Manual Book dengan judul "Teknik Augmentasi Citra untuk Meningkatkan Kinerja Segmentasi menggunakan Deep Learning". Shalawat dan salam tak lupa kita junjungan kepada Nabi kita Rasulullah SAW beserta keluarga, sahabat dan para pengikutnya hingga akhir zaman.

Dalam hal pengimplementasian Artificial Neural Network, penulis telah membangun sistem yang dapat membantu pakar medis dalam klasifikasi secara otomatis jenis penyakit retina. Semoga buku paduan ini dapat bermanfaat dalam klasifikasi otomatis penyakit retina.

Palembang, Februari 2022

Penulis

Erwin

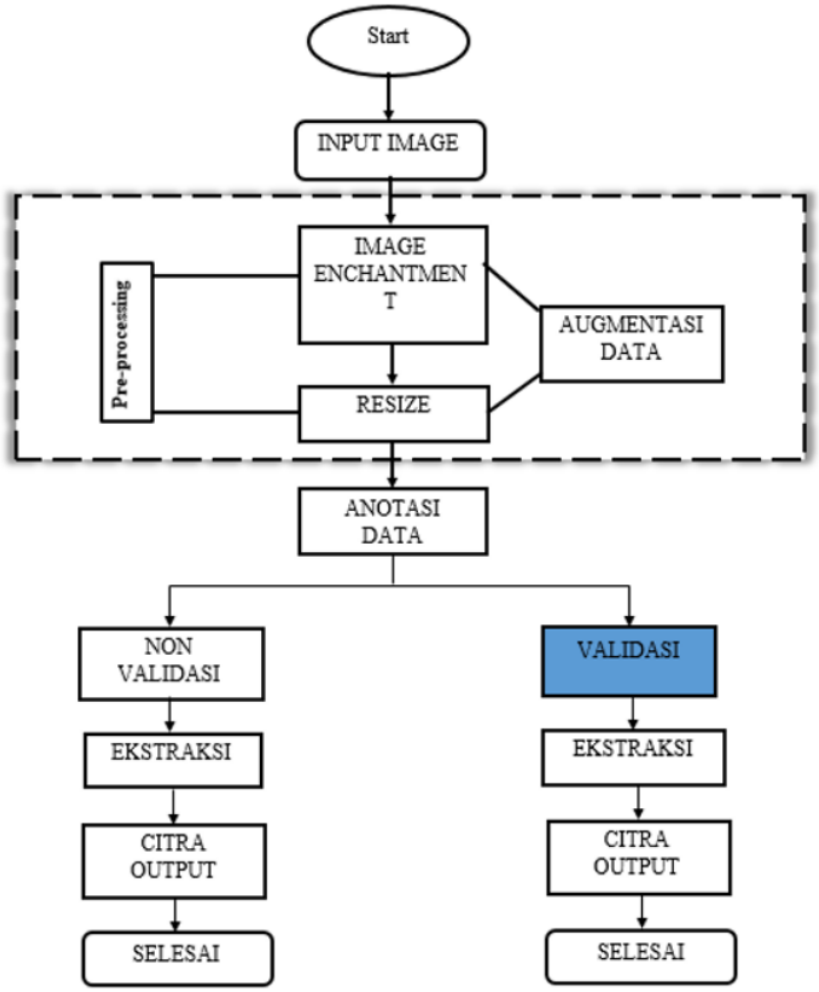
Bambang Suprihatin

1 PENDAHULUAN

1.1. DEKSRIPSI UMUM SISTEM

1. Deskripsi Umum Aplikasi

Dalam proses perancangan sistem pada penelitian ini terdiri menjadi beberapa tahap pemrosesan yaitu input citra, pra-pengolahan, ekstraksi, dan deteksi penyakit citra retina. Dalam penelitian ini dilakukan dengan menggunakan dataset STARE sebanyak 25 citra retina yang terdiri dari 10 penyakit eksudat dan 15 penyakit *diabetic retinopathy*. Tujuannya untuk meningkatkan *precision, mAP* dan *IoU* dan mendapatkan hasil deteksi penyakit pada citra retina lebih baik. Pada tahap perancangan sebelumnya penulis melakukan tahap pra pengolahan seperti *resize* untuk mengubah citra retina, proses selanjutnya image enchantment untuk memperbagus citra retina, proses selanjutnya augmentasi. Augmentasi sendiri tujuannya untuk memperbanyak citra agar semakin banyak varian data citra retina yang digunakan. Setelah dilakukan augmentasi selanjutnya tahap ekstraksi dan deteksi menggunakan *Faster-RCNN* merupakan salah satu langkah penting untuk mengetahui atau mendiagnosa beberapa penyakit yang terkait dengan retina, misalnya mengetahui adanya pertumbuhan pembuluh darah yang tampak pada *optic disk* pada citra retina yang menandai adanya.



Gambar 1. Blok Diagram

2. Deksripsi umum Kebutuhan Aplikasi Yang akan Diimplementasikan

Adapun perangkat keras yang digunakan dalam aplikasi ini yaitu laptop atau PC dengan spesifikasi sebagai berikut:

Prosessor	:	Intel Core i3-4030U, 1.9Ghz.
Memory	:	4GB
System type	:	64-bit Operating System, x64-based processor
OS	:	Windows 10

2 SUMBER DAYA YANG DIBUTUHKAN

2.1 PERANGKAT LUNAK

Perangkat lunak yang digunakan dalam penggunaan aplikasi ini adalah

- Python 3.7
- Spyder

2.2 PERANGKAT KERAS

Perangkat keras yang digunaan dalam penggunaan aplikasi ini adalah:

1. Komputer atau Laptop (offline)
2. Mouse atau bisa juga menggunakan keypad
3. Keyboard sebagai peralatan antarmuka
4. Monitor sebagai peralatan antarmuka

2.3 SUMBER DAYA MANUSIA

Sumber daya manusia yang akan menggunakan aplikasi adalah user yang memiliki pemahaman tentang atarmuka komputer dan cara pengoperasiannya.

3 PENGGUNAAN APLIKASI

3.1 *Input* Citra

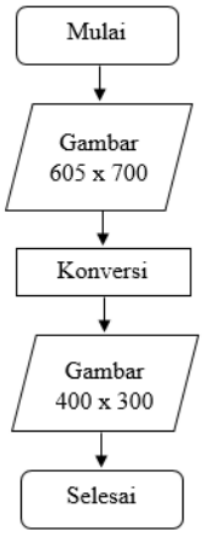
Tahapan awal yang dilakukan adalah mempersiapkan data citra retina sebagai *input* citra. Data citra retina yang digunakan merupakan data yang berasal dari dataset STARE. Dalam penelitian ini data yang digunakan berjumlah 25 citra retina. Basis data yang digunakan pada data STARE memiliki format .ppm. yang dikonversi menjadi jpeg.

3.2. Pra-proses

Pada tahap pra-proses dilakukan untuk memperbaiki kualitas citra retina sehingga dapat membantu pada tahapan ekstraksi dan deteksi. Terdapat beberapa tahapan yang digunakan pada penelitian ini sebagai berikut:

3.3. *Resize* Gambar

Setelah mendapatkan data berupa citra, langkah selanjutnya adalah menyamakan ukuran dimensi pada setiap citra, yang sebelumnya citra tersebut berukuran 605 x 700 *pixel* dan akan dikecilkan menjadi 400 x 300 *pixel*. Ukuran tersebut dipilih untuk menyeimbangi spesifikasi mesin ketika pelatihan data, dimana semakin besar ukuran dimensi citra akan lebih membutuhkan memori. Berikut merupakan tahapan dari proses *resize* gambar dapat dilihat pada gambar 2.



Gambar 2. Tahapan Resize Gambar

3.4. Median Blur

Tahap selanjutnya setelah menkonversi gambar ialah *median blur*. Teknik *median Blur* tahap yang dilakukan pada *image enhancement*. *Median Blur* adalah teknik pemfilteran *digital nonlinear* yang biasa dipakai untuk menghilangkan noise pada citra. Pengurangan *noise* adalah langkah praolah untuk memperbaiki hasil pada pengolahan selanjutnya (misalnya deteksi penyakit pada citra retina). Berikut merupakan *script* dari proses *median blur* dapat dilihat pada gambar 3

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from scipy import ndimage as nd
import glob, os


## path yang digunakan
root_path_testing = 'normal1/*.jpg' ## di path mana gambar akan di blend
root_path_saving = 'medianblurnormal1/'  ## dimana gambar akan disave
file_path = glob.glob(root_path_testing)

#make new directory
os.makedirs(root_path_saving, exist_ok = True)

for path in file_path:
    img=cv2.imread(path)
    kernel = np.ones((3,3),np.float32)/9
    median_blur = nd.median_filter(img, size=3)
    # split filename
    filename = path.split('\\')[-1]
    cv2.imwrite(root_path_saving + filename,median_blur)
```

Gambar 3. Script Proses Median Blur

3.5. Augmentasi Data

Proses selanjutnya ialah mengaugmentasi citra yang berjumlah 25 gambar terdiri dari 15 penyakit *diabetic retinopathy* dan 10 penyakit eksudat. Pada proses augmentasi dilakukan sebanyak lima tahapan, yaitu rotation, brightness, *horizontal flip, zoom range* dan *translation*. Sesuai dengan dataset yang peneliti pakai dari dataset STARE yang berisi citra retina yang telah diaugmentasi. Berikut proses augmentasi yang digunakan untuk penelitian ini sebagai berikut :

3.5.1. Rotation (Rotasi)

Rotation adalah proses perputaran retina pada suatu retina yang tetap. *Rotation* digunakan untuk memutar objek dengan arah putar searah jarum jam atau berlawanan jarum jam. Pada *rotation* peneliti menggunakan *rotate* 20, 30 dan 50 derajat. Berikut merupakan *script* dari proses *rotation* dapat dilihat pada gambar 4.

```
from scipy import ndimage, misc
import imageio
import numpy as np
import os
import cv2

def main():
    outPath = "20EK/"
    path = "EK/"

    # iterate through the names of contents of the folder
    for image_path in os.listdir(path):

        # create the full input path and read the file
        input_path = os.path.join(path, image_path)
        image_to_rotate = imageio.imread(input_path)

        # rotate the image
        rotated = ndimage.rotate(image_to_rotate,  20)
        # create full output path, 'example.jpg'
        # becomes 'rotate_example.jpg', save the file to dis
        fullpath = os.path.join(outPath, '20_'+image_path)
        imageio.imwrite(fullpath, rotated)

if __name__ == '__main__':
    main()
```

Gambar 4. Script Proses Rotation

3.5.2. Brightness (Kecerahan)

Brightness merupakan proses untuk kecerahan pada objek citra, jika intensitas *pixel* dikurangi dengan nilai tertentu maka citra akan menjadi gelap, dan sebaliknya jika intensitas *pixel* ditambah dengan nilai tertentu makan citra akan semakin terang. Pada tingkat kontras *brightness range* peneliti menggunakan 0,2, 0,5 dan 0,8. Berikut merupakan *script* dari proses *brightness* dapat dilihat pada gambar 5.

```
import glob
import os
from layeris.layer_image import LayerImage

## brightness,contrast,gray,etc 1 folder
## path yang digunakan
root_path_testing = 'Before/*.jpg' ## di path mana gambar akan di blend
root_path_saving = 'After/'  ## dimana gambar akan disave
file_path = glob.glob(root_path_testing)

#make new directory
os.makedirs(root_path_saving, exist_ok = True)

for path in file_path:
    image = LayerImage.from_file(path)
    image.grayscale()
    image.brightness(1)
    image.contrast(1)
    image.hue(0.2)
    image.saturation(-0.5)
    image.lightness(-0.8)
    # split filename
    filename = path.split('\\')[-1]
    image.save(root_path_saving + filename,100)
```

Gambar 5. Script Proses Brightness

3.5.3. Flip Horizontal

Flip Horizontal adalah sebuah proses dimana orientasi layar sama dengan *horizontal*, tetapi berbanding terbalik, misalnya citra diatas di *horizontal* menjadi kebawah begitupun dari kiri menjadi kekanan. Berikut merupakan *script* dari proses *flip horizontal* dapat dilihat pada gambar 6.

```
from scipy import ndimage, misc
import imageio
import numpy as np
import os
import cv2

def main():
    outPath = "eksudat/180/"
    path = "eksudat/CItra Asli/"

    # iterate through the names of contents of the folder
    for image_path in os.listdir(path):

        # create the full input path and read the file
        input_path = os.path.join(path, image_path)
        image_to_rotate = imageio.imread(input_path)

        # rotate the image
        rotated = ndimage.rotate(image_to_rotate, 180)

        # create full output path, 'example.jpg'
        # becomes 'rotate_example.jpg', save the file to disk
        fullpath = os.path.join(outPath, '180_'+image_path)
        imageio.imwrite(fullpath, rotated)

if __name__ == '__main__':
    main()
```

Gambar 6. Script Proses Flip Horizontal

3.5.4. Zoom Range (Rentang Pendekatan)

Zoom Range adalah suatu proses yang memainkan jarak citra dengan tujuan memperbesar tampilan objek dengan cara mendekatkan citra dan memperkecil tampilan objek pada citra. Pada *zoom range* peneliti memainkan jarak citra 0,7-1,0. Berikut merupakan *script* dari proses *zoom range* dapat dilihat pada gambar 7.

```
import imageio
import imgaug as ia
import imgaug.augmenters as iaa
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib
import cv2
import glob
import os

#Clahe 1 folder
root_path_testing = 'eksudat/ca/*.jpg'
root_path_saving = 'eksudat/ca/zoom/'
file_path = glob.glob(root_path_testing)


#make new directory
os.makedirs(root_path_saving, exist_ok = True)

#image enchantment using clahe

for path in file_path:
    image = cv2.imread(path, 1)
    scale_im=iaa.Affine(scale={"x": (0.7, 1.0), "y": (0.7, 1.0)})
    scale_image =scale_im.augment_image(image)

    # split filename
    filename = path.split('\\')[-1]
    cv2.imwrite(root_path_saving + filename, scale_image )
```

Gambar 7. Script Proses Zoom Range

3.5.5. Translation (Translasi)

Translation adalah suatu pergerakkan/perpindahan semua titik dari objek pada suatu jalur lurus sehingga menempati posisi baru, artinya objek sebelumnya berubah ke posisi yang baru. Pada *translation range* peneliti menggerakan objek 0,2-0,01. Berikut merupakan *script* dari proses *translation* dapat dilihat pada gambar 8..

```
#Clahe 1 folder
root_path_testing = 'diabetic/real/*.png'
root_path_saving = 'diabetic/translating/'
file_path = glob.glob(root_path_testing)


#make new directory
os.makedirs(root_path_saving, exist_ok = True)

#image enchantment using clahe

M = np.float32([[1, 0, 20], [0, 1, 10]])

for path in file_path:
    image = cv2.imread(path, 1)




    # Read image from disk.
    (rows, cols) = image.shape[:2]

    # warpAffine does appropriate shifting given the
    # translation matrix.
    res = cv2.warpAffine(image, M, (cols, rows))

    # split filename
    filename = path.split('\\')[-1]
    cv2.imwrite(root_path_saving + filename, res )
```

Gambar 8. Script Proses Translation

3.6. Anotasi Data

Setelah tahap augmentasi, pada tahap ini melakukan anotasi data, bagian yang akan dianotasi adalah penyakit pada retina mata. Proses anotasi ini dilakukan dengan cara manual dengan bantuan aplikasi *labellmg* yang disediakan *python*. Bagian yang diambil terdiri dari penyakit *diabetic retinopathy* dan eksudat. Anotasi data dilakukan untuk tahap pelatihan dan evaluasi. Hasil yang didapatkan dari anotasi data berupa format xml. Setelah mendapatkan hasil anotasi penulis dapat melakukan proses pelatihan *(Training)*. Berikut merupakan proses label manual pada gambar 9.

Gambar 9, Tahapan Proses Anotasi Data

3.7. Fitur Ekstraksi

Tahapan selanjutnya adalah fitur ekstraksi. Fitur ekstraksi atau biasanya disebut dengan *backbone* merupakan proses pengenalan ciri karakteristik. Proses ini membuat jaringan dapat saling terhubung, *backbone* yang digunakan yaitu *ResNet-50*. Pada penelitian ini penulis menggunakan *ResNet-50*. *ResNet-50* adalah salah satu varian *ResNet* sebelumnya dilakukan *skip connection* sebanyak 2 *layer*, maka *ResNet-50* melewati 3 *layer* dan terdapat 1x1 *convolution layer*. Pada proses Konvolusi terjadi 5 kali. Proses ukuran keluaran *(output size)* dari tinggi ke rendah, menggunakan 2 *stride*, *max pooling* 3 x 3 dan lapisan terakhir *(fully connected)* menggunakan aktifasi *softmax*. Konvolusi *ResNet-50* tersebut dapat dilihat tabel 1.

Tabel 1 Konvolusi ResNet-50

Layer Name	Output Size	50 layer
Conv 1	112 x 112	7x7, 64 Stride 2
Conv 2	56 x 56	3 x 3 max pool, stride 2
Conv 3	28 x 28	$\begin{bmatrix} 1 \times 1,64 \\ 3 \times 3,64 \\ 1 \times 1,256 \end{bmatrix} \times 3$
Conv 4	14 x 14	$\begin{bmatrix} 1 \times 1,256 \\ 3 \times 3,256 \\ 1 \times 1,1024 \end{bmatrix} \times 6$
Conv 5	7 x 7	$\begin{bmatrix} 1 \times 1,512 \\ 3 \times 3,512 \\ 1 \times 1,2048 \end{bmatrix} \times 3$
	1 x 1	Average pool , 1000-d FC, Softmax
FLOPs		$3{,}8 \times 10^9$

3.8. Deteksi Penyakit Pada Citra Retina

Tahap selanjutnya ialah deteksi. Deteksi adalah mengenali suatu objek pada citra atau pengenalan dilakukan melalui ciri-ciri penyakit yang dimiliki pada citra retina.. Deteksi pada penelitian ini menggunakan arsitektur *ResNet-50* untuk non validasi yang akan didukung dengan beberapa parameter sebagai berikut *Epoch*, *Batch Size*, *Input Layer*, dan *Learning Rate*. Untuk parameter kedua, yaitu yang di validasi juga menggunakan parameter yang sama. Parameter tersebut dapat dilihat pada tabel dibawah ini.

Tabel 2 Parameter Arsitektur ResNet-50 Model 1

Model Parameter	Isi Parameter
Epoch	1000
Batch Size	32
Input Layer	256.256.3
Learning Rate	0.001

Tabel 3. Parameter Arsitektur ResNet-50 Model 2

Model Parameter	Isi Parameter
Epoch	1500
Batch Size	64
Input Layer	256.256.3
Learning Rate	0.001

3.9. Evaluasi

Pada tahapan ini, hasil citra penyakit retina yang sudah dibangun akan dilakukan proses evaluasi dengan tujuan untuk mengukur performa dari arsitektur dan dataset yang diusulkan. Parameter yang digunakan untuk mendapatkan hasil evaluasi ini yaitu *Precision, Map* dan *IoU*.

3.10. Hasil Program Model Faster-RCNN RESNET-50

```
[ ]  # install dependencies: (use cu101 because colab has CUDA 10.1)
     !pip install -U torch==1.5 torchvision==0.6 -f https://download.pytorch.org/whl/cu101/torch_stable.html
     !pip install cython pyyaml==5.1
     !pip install -U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'
     import torch, torchvision
     print(torch.__version__, torch.cuda.is_available())
     !gcc --version
     # opencv is pre-installed on colab
```

```
[ ]  # install detectron2:
     !pip install detectron2==0.1.3 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.5/index.html
```

Install enviroment di google colab agar nantinya bisa menggunakan fungsi detectron 2

```
[ ]  # You may need to restart your runtime prior to this, to let your installation take effect
     # Some basic setup:
     # Setup detectron2 logger
     import detectron2
     from detectron2.utils.logger import setup_logger
     setup_logger()

     # import some common libraries
     import numpy as np
     import cv2
     import random
     from google.colab.patches import cv2_imshow

     # import some common detectron2 utilities
     from detectron2 import model_zoo
     from detectron2.engine import DefaultPredictor
     from detectron2.config import get_cfg
     from detectron2.utils.visualizer import Visualizer
     from detectron2.data import MetadataCatalog
     from detectron2.data.catalog import DatasetCatalog
```

Import library yang akan digunakan

```
[ ]  !curl -L "https://app.roboflow.com/ds/XrJkXolDTu?key=S4tt4TfxvD" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

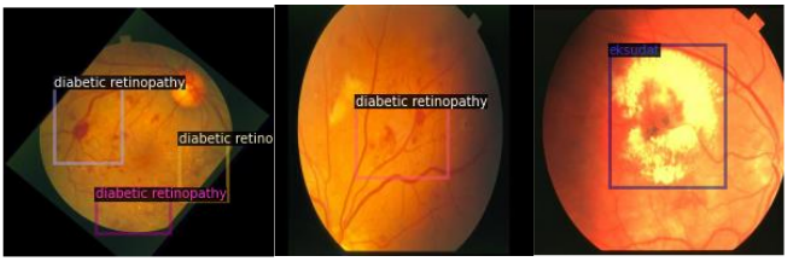
Import data yang sudah kita siapkan dari roboflow ke goolge colab

```
[ ]  from detectron2.data.datasets import register_coco_instances
     register_coco_instances("my_dataset_train", {}, "/content/train/_annotations.coco.json", "/content/train")
     register_coco_instances("my_dataset_val", {}, "/content/valid/_annotations.coco.json", "/content/valid")
     register_coco_instances("my_dataset_test", {}, "/content/test/_annotations.coco.json", "/content/test")
```

Mengatur path atau tujuan dataset, data train data test

```
[ ]  #visualize training data
     my_dataset_train_metadata = MetadataCatalog.get("my_dataset_train")
     dataset_dicts = DatasetCatalog.get("my_dataset_train")

     import random
     from detectron2.utils.visualizer import Visualizer

     for d in random.sample(dataset_dicts, 3):
         img = cv2.imread(d["file_name"])
         visualizer = Visualizer(img[:, :, ::-1], metadata=my_dataset_train_metadata, scale=0.5)
         vis = visualizer.draw_dataset_dict(d)
         cv2_imshow(vis.get_image()[:, :, ::-1])
```

Visualisasi contoh gambar dan anotasi manual




```
[ ] #We are importing our own Trainer Module here to use the COCO validation evaluation during training. Otherwise no validation eval occurs.

    from detectron2.engine import DefaultTrainer
    from detectron2.evaluation import COCOEvaluator

    class CocoTrainer(DefaultTrainer):

      @classmethod
      def build_evaluator(cls, cfg, dataset_name, output_folder=None):

        if output_folder is None:
            os.makedirs("coco_eval", exist_ok=True)
            output_folder = "coco_eval"

        return COCOEvaluator(dataset_name, cfg, False, output_folder)
```

Membuat fungsi untuk menghitung evaluasi

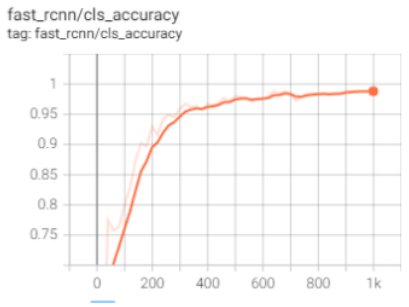
```
[ ] #from .detectron2.tools.train_net import Trainer
    #from detectron2.engine import DefaultTrainer
    # select from modelzoo here: https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md#coco-object-detection-baselines

    from detectron2.config import get_cfg
    #from detectron2.evaluation.coco_evaluation import COCOEvaluator
    import os

    cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_DC5_3x.yaml"))
    cfg.DATASETS.TRAIN = ("my_dataset_train",)
    cfg.DATASETS.TEST = ("my_dataset_val",)

    cfg.DATALOADER.NUM_WORKERS = 4
    cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_DC5_3x.yaml")  # Let training initialize from model zoo
    cfg.SOLVER.IMS_PER_BATCH = 4
    cfg.SOLVER.BASE_LR = 0.001 #LR


    cfg.SOLVER.WARMUP_ITERS = 500 # epoch
    cfg.SOLVER.MAX_ITER = 1000 #adjust up if val mAP is still rising, adjust down if overfit
    cfg.SOLVER.STEPS = (1000, 1500)
    cfg.SOLVER.GAMMA = 0.05



    cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64        # batch size
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 3 #your number of classes + 1
```

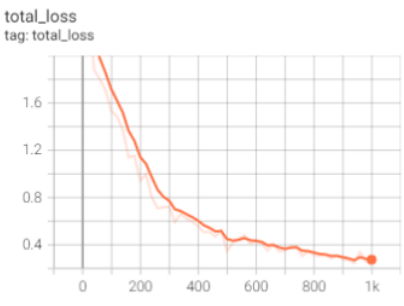
```
cfg.TEST.EVAL_PERIOD = 500


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

Melakukan train dan testing dengan menggunakan epoch 500, learning rate 0,001 dan batch size 64

```
[ ]  # Look at training curves in tensorboard:
     %load_ext tensorboard
     %tensorboard --logdir output
```

Membuat dengan tensorboard untuk mendapatkan hasil total accuracy dan total loss






```
#test evaluation
from detectron2.data import DatasetCatalog, MetadataCatalog, build_detection_test_loader
from detectron2.evaluation import COCOEvaluator, inference_on_dataset

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.85
predictor = DefaultPredictor(cfg)
evaluator = COCOEvaluator("my_dataset_test", cfg, False, output_dir="./output/")
val_loader = build_detection_test_loader(cfg, "my_dataset_test")
inference_on_dataset(trainer.model, val_loader, evaluator)
```

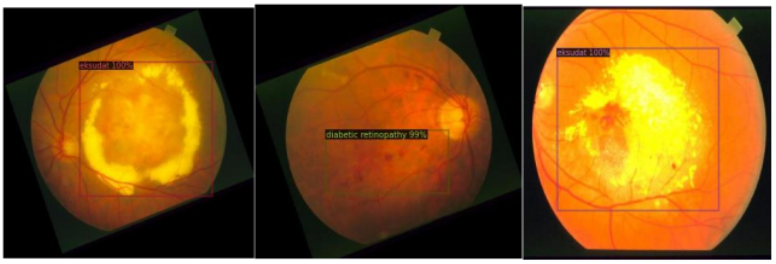
```
category    | AP    | category              | AP      | category
| AP      |
|:-----------|:-----|:---------------------|:-------|:--------
---|:-------|
| traffic     | nan   | diabetic retinopathy | 81.683 | eksudat
| 95,050 | mean average precision | 88,366 | IoU = 81,7
```

```
[ ]  cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
     cfg.DATASETS.TEST = ("my_dataset_test", )
     cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set the testing threshold for this model
     predictor = DefaultPredictor(cfg)
     test_metadata = MetadataCatalog.get("my_dataset_test")
```

Save model kebentuk pth

```
[ ]  from detectron2.utils.visualizer import ColorMode
     import glob

     for imageName in glob.glob('/content/test/*jpg'):
       im = cv2.imread(imageName)
       outputs = predictor(im)
       v = Visualizer(im[:, :, ::-1],
                      metadata=test_metadata,
                      scale=0.8
                       )
       out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
       cv2_imshow(out.get_image()[:, :, ::-1])
```

Visualisasi gambar dengan mesin



Teknik Augmentasi Citra untuk Meningkatkan Kinerja Segmentasi menggunakan Deep Learning

ORIGINALITY REPORT

25%	24%	8%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.bsi.ac.id Internet Source	2%
2	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Student Paper	2%
3	publisher.uthm.edu.my Internet Source	2%
4	ejurnal.amikstiekomsu.ac.id Internet Source	2%
5	jmai.mercubuana-yogya.ac.id Internet Source	2%
6	id.wikipedia.org Internet Source	2%
7	repository.upnvj.ac.id Internet Source	1%
8	id.123dok.com Internet Source	1%
	arifrizaazizi.blogspot.com	

9	Internet Source	1 %
10	fr.scribd.com Internet Source	1 %
11	doku.pub Internet Source	1 %
12	nikoprasetia.wordpress.com Internet Source	1 %
13	docplayer.info Internet Source	1 %
14	Submitted to Universitas Muhammadiyah Surakarta Student Paper	1 %
15	apptopfan.com Internet Source	1 %
16	jurnal.upnyk.ac.id Internet Source	1 %
17	Chuan Dai, Yajuan Wei, Zhijie Xu, Minsi Chen, Ying Liu, Jiulun Fan. "An Investigation into Performance Factors of Two-Stream I3D Networks", 2021 26th International Conference on Automation and Computing (ICAC), 2021 Publication	1 %
18	anakkampungsidikalang.blogspot.com Internet Source	

		1 %
19	jurnal.polban.ac.id Internet Source	1 %
20	text-id.123dok.com Internet Source	1 %
21	dantiambarwati.blogspot.com Internet Source	1 %
22	repository.its.ac.id Internet Source	1 %
23	repository.usu.ac.id Internet Source	1 %

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	Off		