DAFTAR PUSTAKA

[1] G. Perveen, M. Rizwan, N. Goel, and P. Anand, “Artificial neural network models for global solar energy and photovoltaic power forecasting over India,” *Energy Sources, Part A: Recovery, Utilization and Environmental Effects*, vol. 00, no. 00, pp. 1–26, 2020, doi: 10.1080/15567036.2020.1826017.

[2] H. Gunerhan, A. Hepbasli, and U. Giresunlu, “Environmental impacts from the solar energy systems,” *Energy Sources, Part A: Recovery, Utilization and Environmental Effects*, vol. 31, no. 2, pp. 1131–1138, 2009, doi: 10.1080/15567030701512733.

[3] I. D’Adamo, M. Gastaldi, and P. Morone, “The post COVID-19 green recovery in practice: Assessing the profitability of a policy proposal on residential photovoltaic plants,” *Energy Policy*, vol. 147, no. September, p. 111910, 2020, doi: 10.1016/j.enpol.2020.111910.

[4] W. Tang, Q. Yang, K. Xiong, and W. Yan, “Deep learning based automatic defect identification of photovoltaic module using electroluminescence images,” *Solar Energy*, vol. 201, no. March, pp. 453–460, 2020, doi: 10.1016/j.solener.2020.03.049.

[5] D. S. Pillai and N. Rajasekar, “A comprehensive review on protection challenges and fault diagnosis in PV systems,” *Renewable and Sustainable Energy Reviews*, vol. 91, no. July 2017, pp. 18–40, 2018, doi: 10.1016/j.rser.2018.03.082.

[6] D. S. Pillai and N. Rajasekar, “A comprehensive review on protection challenges and fault diagnosis in PV systems,” *Renewable and Sustainable Energy Reviews*, vol. 91. Elsevier Ltd, pp. 18–40, Aug. 01, 2018. doi: 10.1016/j.rser.2018.03.082.

[7] S. Meyer *et al.*, “Silver nanoparticles cause snail trails in photovoltaic modules,” *Solar Energy Materials and Solar Cells*, vol. 121, pp. 171–175, 2014, doi: https://doi.org/10.1016/j.solmat.2013.11.013.

[8] H.-C. Liu, C.-T. Huang, W.-K. Lee, S.-S. Yan, and F.-M. Lin, “A Defect Formation as Snail Trails in Photovoltaic Modules,” *Energy Power Eng*, vol. 07, no. 08, pp. 348–353, 2015, doi: 10.4236/epe.2015.78032.

[9] N. Kim, K. J. Hwang, D. Kim, J. H. Lee, S. Jeong, and D. H. Jeong, “Analysis and reproduction of snail trails on silver grid lines in crystalline silicon photovoltaic modules,” *Solar Energy*, vol. 124, pp. 153–162, Feb. 2016, doi: 10.1016/j.solener.2015.11.040.

[10] M. Köntges, S. Kajari-Schröder, and I. Kunze, “Crack statistic for wafer-based silicon solar cell modules in the field measured by UV fluorescence,”

IEEE J Photovolt, vol. 3, no. 1, pp. 95–101, 2013, doi: 10.1109/JPHOTOV.2012.2208941.

[11] A. Morlier, F. Haase, and M. Kontges, "Impact of Cracks in Multicrystalline Silicon Solar Cells on PV Module Power -A Simulation Study Based on Field Data," *IEEE J Photovolt*, vol. 5, no. 6, pp. 1735–1741, Sep. 2015, doi: 10.1109/JPHOTOV.2015.2471076.

[12] J. Yu, Z. Wang, A. Majumdar, and R. Rajagopal, "DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States," *Joule*, vol. 2, no. 12, pp. 2605–2617, Dec. 2018, doi: 10.1016/j.joule.2018.11.021.

[13] C. Zhang, C. Wen, and J. Liu, "Mask-MRNet: A deep neural network for wind turbine blade fault detection," *Journal of Renewable and Sustainable Energy*, vol. 12, no. 5, Sep. 2020, doi: 10.1063/5.0014223.

[14] A. Shihavuddin *et al.*, "Wind Turbine Surface Damage Detection by Deep Learning Aided Drone Inspection Analysis," 2019, doi: 10.17632/hd96prn3nc.1.

[15] Y. Zefri, A. Elkettani, I. Sebari, and S. A. Lamallam, "Thermal infrared and visual inspection of photovoltaic installations by uav photogrammetry—application case: Morocco," *Drones*, vol. 2, no. 4, pp. 1–24, Dec. 2018, doi: 10.3390/drones2040041.

[16] IEEE Staff, *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC).* IEEE, 2016.

[17] W. Kean Yap, R. Galet, and K. C. Yeo, "Quantitative Analysis of Dust and Soiling on Solar PV Panels in the Tropics Utilizing Image-Processing Methods."

[18] H. Chen, Y. Pang, Q. Hu, and K. Liu, "Solar cell surface defect inspection based on multispectral convolutional neural network," *J Intell Manuf*, vol. 31, no. 2, pp. 453–468, Feb. 2020, doi: 10.1007/s10845-018-1458-z.

[19] M. R. Maghami, H. Hizam, C. Gomes, M. A. Radzi, M. I. Rezadad, and S. Hajighorbani, "Power loss due to soiling on solar panel: A review," *Renewable and Sustainable Energy Reviews*, vol. 59. Elsevier Ltd, pp. 1307–1316, Jun. 01, 2016. doi: 10.1016/j.rser.2016.01.044.

[20] Anne. Labouret, Michel. Villoz, and Impr. Chirat), *Énergie solaire photovoltaïque*. Dunod, 2009.

[21] S. R. Madeti and S. N. Singh, "Monitoring system for photovoltaic plants: A review," *Renewable and Sustainable Energy Reviews*, vol. 67. Elsevier Ltd, pp. 1180–1207, Jan. 01, 2017. doi: 10.1016/j.rser.2016.09.088.

[22] W. I. Ley, "Electricity from Sunlight."

[23] C. Ferrara and D. Philipp, “Why do PV modules fail?,” in *Energy Procedia*, 2012, vol. 15, pp. 379–387. doi: 10.1016/j.egypro.2012.02.046.

[24] M. 1972- Köntges *et al.*, *Performance and reliability of photovoltaic systems subtask 3.2: Review of failures of photovoltaic modules : IEA PVPS task 13 : external final report IEA-PVPS*.

[25] Y. Wang, K. Itako, T. Kudoh, K. Koh, and Q. Ge, “Voltage-based hot-spot detection method for photovoltaic string using a projector,” *Energies (Basel)*, vol. 10, no. 2, 2017, doi: 10.3390/en10020230.

[26] M. Afridi *et al.*, “Determining the Effect of Soiling and Dirt Particles at Various Tilt Angles of Photovoltaic Modules,” 2017. [Online]. Available: www.kwpublisher.com

[27] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015. doi: 10.1038/nature14539.

[28] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*, vol. 61. Elsevier Ltd, pp. 85–117, Jan. 01, 2015. doi: 10.1016/j.neunet.2014.09.003.

[29] Y. Bengio, “Learning deep architectures for AI,” *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009, doi: 10.1561/2200000006.

[30] W. S. Eka Putra, “Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101,” *Jurnal Teknik ITS*, vol. 5, no. 1, 2016, doi: 10.12962/j23373539.v5i1.15696.

[31] C. Liao, H. Chen, Y. Sun, and H. Ding, “An Analysis of Single-Layer Networks in Unsupervised Feature Learning Adam,” *Qiche Gongcheng/Automotive Engineering*, vol. 24, no. 2, p. 130, 2002.

[32] K. Fukushima, “Biological Cybernetics,” vol. 202, 1980.

[33] Y. U. Hanafi *et al.*, “DETEKSI PENGGUNAAN HELM PADA PENGENDARA BERMOTOR BERBASIS DEEP LEARNING.”

[34] R. Sinaga, J. Tanesab, and M. Beily, “The Impact of Snail Trails and Cracks to Energy Output of Photovoltaic Modules at Lotas Off-Grid PV System,” Dec. 2019. doi: 10.4108/eai.18-10-2019.2289995.

[35] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *Proceedings of the 32nd International Conference on Machine Learning*, Aug. 2015, vol. 37, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html

[36] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, “Land Use Classification in Remote Sensing Images by Convolutional Neural

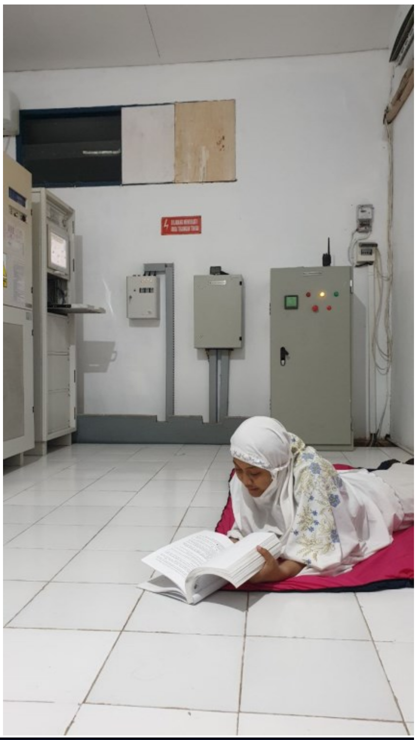
Networks," Aug. 2015, [Online]. Available: http://arxiv.org/abs/1508.00092

[37] M. B. Bejiga, A. Zeggada, A. Nouffidj, and F. Melgani, "A convolutional neural network approach for assisting avalanche search and rescue operations with UAV imagery," *Remote Sens (Basel)*, vol. 9, no. 2, 2017, doi: 10.3390/rs9020100.

[38] S. Albelwi and A. Mahmood, "A framework for designing the architectures of deep Convolutional Neural Networks," *Entropy*, vol. 19, no. 6, Jun. 2017, doi: 10.3390/e19060242.

LAMPIRAN

Dokumentasi Pengambilan Data



Source Code Training YOLOv3 dengan Framework Darknet

```
#Command untuk mendownload model pre-train YOLO yang akan
digunakan untuk proses training
!git clone https://github.com/AlexeyAB/darknet
```

Cloning into 'darknet'...
remote: Enumerating objects: 15502, done.
remote: Total 15502 (delta 0), reused 0 (delta 0), pack-reused 15502
Receiving objects: 100% (15502/15502), 14.17 MiB | 27.44 MiB/s, done.
Resolving deltas: 100% (10403/10403), done.
KodeTeks

```
#Command untuk mengkases drive folder
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')

# Command untuk membuat tautan simbolis sehingga sekarang
jalur /content/gdrive/My\Drive/ sama dengan /mydrive
!ln -s /content/gdrive/My\ Drive/ /mydrive

# Command untuk mengecek konten pada folder yolov3
!ls /mydrive/yolov3
```

```
Mounted at /content/gdrive
obj.data     obj.names     obj.zip     process.py     training
yolov3custom.cfg
```

```
# Command untuk mengubah makefile agar dapat diaktifkan ke GPU
dan OpenCV
# Command untuk mengatur CUDNN, CUDNN_HALF dan LIBSO ke 1

%cd /content/darknet/
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile
/content/darknet

# Command untuk membangun darknet
!make
mkdir -p ./obj/
mkdir -p backup
chmod +x *.sh


# Command untuk membersihkan folder data dan cfg terlebih
dahulu kecuali folder label di data yang diperlukan

%cd data/
!find -maxdepth 1 -type f -exec rm -rf {} \;
%cd ..

%rm -rf cfg/
%mkdir cfg
/content/darknet/data
/content/darknet

#Command untuk menyaalin file zip dataset ke folder root
darknet
!cp /mydrive/yolov3/obj.zip ../

# Command untuk meng unzip kumpulan dataset dan isinya
sehingga sekarang berada di folder /darknet/data/
!unzip ../obj.zip -d data/


Archive:  ../obj.zip
  inflating: data/obj/DJI_0306.jpg
  inflating: data/obj/DJI_0306.txt
  inflating: data/obj/DJI_0307.jpg
  inflating: data/obj/DJI_0307.txt
  inflating: data/obj/DJI_0308.jpg
  inflating: data/obj/DJI_0308.txt
  inflating: data/obj/DJI_0309.jpg
  inflating: data/obj/DJI_0309.txt
  inflating: data/obj/DJI_0310.jpg
  inflating: data/obj/DJI_0310.txt
  inflating: data/obj/DJI_0311.jpg
 extracting: data/obj/DJI_0311.txt
  inflating: data/obj/DJI_0312.jpg
```

```
# Command untuk menyalin file cfg khusus dari drive ke folder darknet/cfg
!cp /mydrive/yolov3/yolov3custom.cfg ./cfg


# Command untuk menyalin file obj.names dan obj.data sehingga sekarang berada di folder /darknet/data/
!cp /mydrive/yolov3/obj.names ./data
!cp /mydrive/yolov3/obj.data  ./data


# Perintah untuk menyalin file process.py dari drive ke direktori darknet
!cp /mydrive/yolov3/process.py ./


# Command untuk menjalankan process.py (perintah ini untuk membuat file train.txt dan test.txt di folder darknet/data yang telah dibuat)
!python process.py

# Command untuk mengecek daftar isi folder data untuk memeriksa apakah file train.txt dan test.txt telah dibuat
!ls data/


/content/darknet
labels     obj  obj.data  obj.names  test.txt  train.txt

# Command untuk men download the yolov3 pre-trained weights file
!wget https://pjreddie.com/media/files/darknet53.conv.74
--2023-01-06 16:37:05--  https://pjreddie.com/media/files/darknet53.conv.74

Resolving pjreddie.com (pjreddie.com)... 128.208.4.108

Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 162482580 (155M) [application/octet-stream]

Saving to: ‘darknet53.conv.74’



darknet53.conv.74   100%[===================>] 154.96M  16.9MB/s    in 10s



2023-01-06 16:37:16 (15.4 MB/s) - ‘darknet53.conv.74’ saved [162482580/162482580]

# Command untuk melakukan training menggunakan file konfigurasi snailtrails.cfg
```

```
# %%capture

!./darknet detector train data/obj.data cfg/yolov3custom.cfg d
arknet53.conv.74 -dont_show -map


# define helper function imShow
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
  %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interp
olation = cv2.INTER_CUBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  #plt.show('')
```

***Source Code Testing Snail Trails* dengan YOLOv3**

```
# Command untuk melakukan training

!./darknet detector test data/obj.data cfg/yolov3custom.cfg /m
ydrive/yolov3/training/yolov3custom_best.weights /mydrive/imag
es/DJI_0265.jpg -thresh 0.3
imShow('predictions.jpg')


CUDA-version: 11020 (11020), cuDNN: 8.1.1, CUDNN_HALF=1, GPU
count: 1
 CUDNN_HALF=1
 OpenCV version: 3.2.0
 0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4
net.optimized_memory = 0
mini_batch = 1, batch = 1, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input
output
```

```
   0 Create CUDA-stream - 0
 Create cudnn-handle 0
conv     32       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x
32 0.299 BF
   1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x
208 x  64 1.595 BF
   2 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x
208 x  32 0.177 BF
   3 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x
208 x  64 1.595 BF
   4 Shortcut Layer: 1,  wt = 0, wn = 0, outputs: 208 x 208 x
64 0.003 BF
   5 conv    128       3 x 3/ 2    208 x 208 x  64 ->  104 x
104 x 128 1.595 BF
   6 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x
104 x  64 0.177 BF
   7 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x
104 x 128 1.595 BF
   8 Shortcut Layer: 5,  wt = 0, wn = 0, outputs: 104 x 104 x
128 0.001 BF
   9 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x
104 x  64 0.177 BF
  10 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x
104 x 128 1.595 BF
  11 Shortcut Layer: 8,  wt = 0, wn = 0, outputs: 104 x 104 x
128 0.001 BF
  12 conv    256       3 x 3/ 2    104 x 104 x 128 ->   52 x
52 x 256 1.595 BF
  13 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
  14 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  15 Shortcut Layer: 12,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  16 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
  17 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  18 Shortcut Layer: 15,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  19 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
  20 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  21 Shortcut Layer: 18,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  22 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
  23 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  24 Shortcut Layer: 21,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  25 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
```

```
  26 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  27 Shortcut Layer: 24,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  28 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
  29 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  31 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
  32 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  33 Shortcut Layer: 30,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  34 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
  35 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
  36 Shortcut Layer: 33,  wt = 0, wn = 0, outputs:  52 x  52 x
256 0.001 BF
  37 conv    512       3 x 3/ 2     52 x  52 x 256 ->   26 x
26 x 512 1.595 BF
  38 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  39 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  40 Shortcut Layer: 37,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
  41 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  42 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  43 Shortcut Layer: 40,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
  44 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  45 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  46 Shortcut Layer: 43,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
  47 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  48 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  49 Shortcut Layer: 46,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
  50 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  51 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  52 Shortcut Layer: 49,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
```

```
  53 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  54 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  55 Shortcut Layer: 52,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
  56 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  57 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  58 Shortcut Layer: 55,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
  59 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  60 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  61 Shortcut Layer: 58,  wt = 0, wn = 0, outputs:  26 x  26 x
512 0.000 BF
  62 conv   1024       3 x 3/ 2     26 x  26 x 512 ->   13 x
13 x1024 1.595 BF
  63 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x 512 0.177 BF
  64 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x
13 x1024 1.595 BF
  65 Shortcut Layer: 62,  wt = 0, wn = 0, outputs:  13 x  13
x1024 0.000 BF
  66 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x 512 0.177 BF
  67 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x
13 x1024 1.595 BF
  68 Shortcut Layer: 65,  wt = 0, wn = 0, outputs:  13 x  13
x1024 0.000 BF
  69 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x 512 0.177 BF
  70 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x
13 x1024 1.595 BF
  71 Shortcut Layer: 68,  wt = 0, wn = 0, outputs:  13 x  13
x1024 0.000 BF
  72 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x 512 0.177 BF
  73 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x
13 x1024 1.595 BF
  74 Shortcut Layer: 71,  wt = 0, wn = 0, outputs:  13 x  13
x1024 0.000 BF
  75 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x 512 0.177 BF
  76 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x
13 x1024 1.595 BF
  77 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x 512 0.177 BF
  78 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x
13 x1024 1.595 BF
  79 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x 512 0.177 BF
```

```
  80 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x
13 x1024 1.595 BF
  81 conv     21       1 x 1/ 1     13 x  13 x1024 ->   13 x
13 x  21 0.007 BF
  82 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm:
1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
  83 route  79                                 ->   13 x  13
x 512
  84 conv    256       1 x 1/ 1     13 x  13 x 512 ->   13 x
13 x 256 0.044 BF
  85 upsample                2x    13 x  13 x 256 ->   26 x
26 x 256
  86 route  85 61                              ->   26 x  26
x 768
  87 conv    256       1 x 1/ 1     26 x  26 x 768 ->   26 x
26 x 256 0.266 BF
  88 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  89 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  90 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  91 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x 256 0.177 BF
  92 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x
26 x 512 1.595 BF
  93 conv     21       1 x 1/ 1     26 x  26 x 512 ->   26 x
26 x  21 0.015 BF
  94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm:
1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
  95 route  91                                 ->   26 x  26
x 256
  96 conv    128       1 x 1/ 1     26 x  26 x 256 ->   26 x
26 x 128 0.044 BF
  97 upsample                2x    26 x  26 x 128 ->   52 x
52 x 128
  98 route  97 36                              ->   52 x  52
x 384
  99 conv    128       1 x 1/ 1     52 x  52 x 384 ->   52 x
52 x 128 0.266 BF
 100 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
 101 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
 102 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
 103 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x 128 0.177 BF
 104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x
52 x 256 1.595 BF
 105 conv     21       1 x 1/ 1     52 x  52 x 256 ->   52 x
52 x  21 0.029 BF
```

```
 106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm:
1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.312
avg_outputs = 516922
 Allocate additional workspace_size = 52.44 MB
Loading weights from
/mydrive/yolov3/training/yolov3custom_best.weights...
 seen 64, trained: 147 K-images (2 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
 Detection layer: 82 - type = 28
 Detection layer: 94 - type = 28
 Detection layer: 106 - type = 28
/mydrive/images/DJI_0265.jpg: Predicted in 30.393000 milli-
seconds.
```