

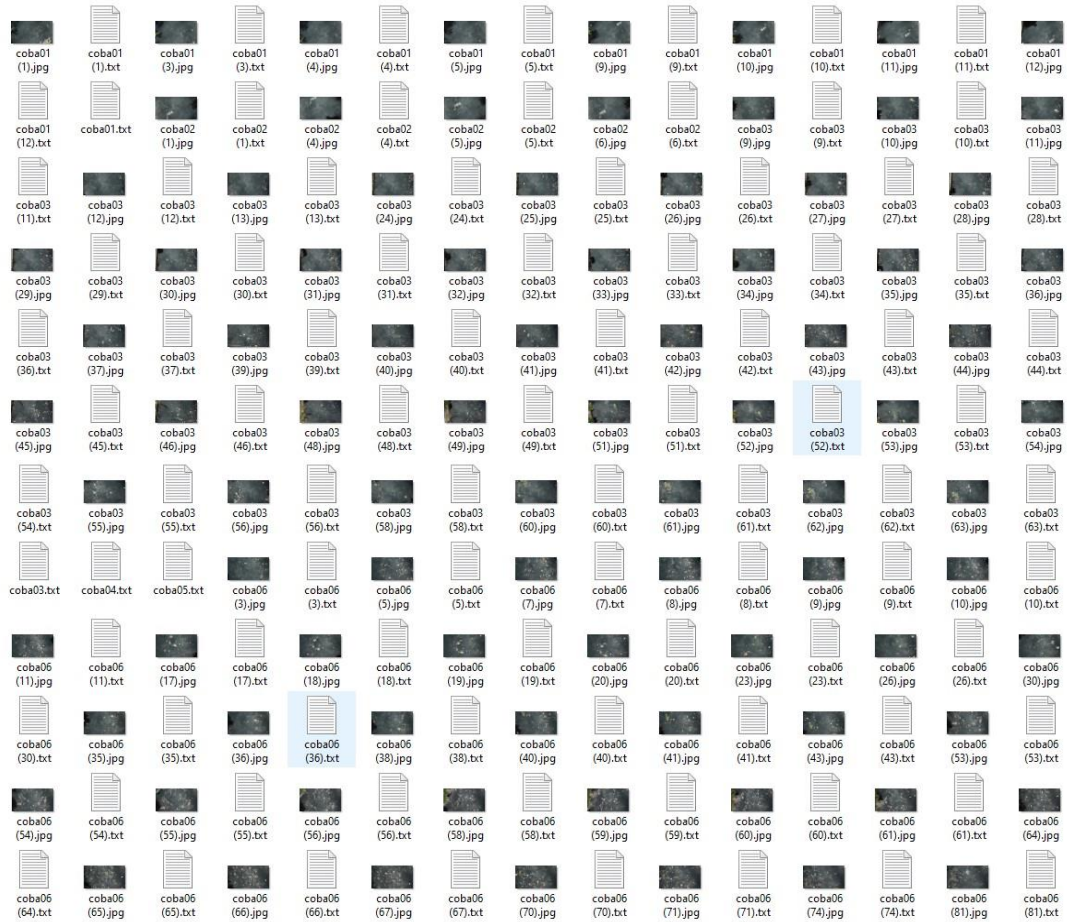
## DAFTAR PUSTAKA

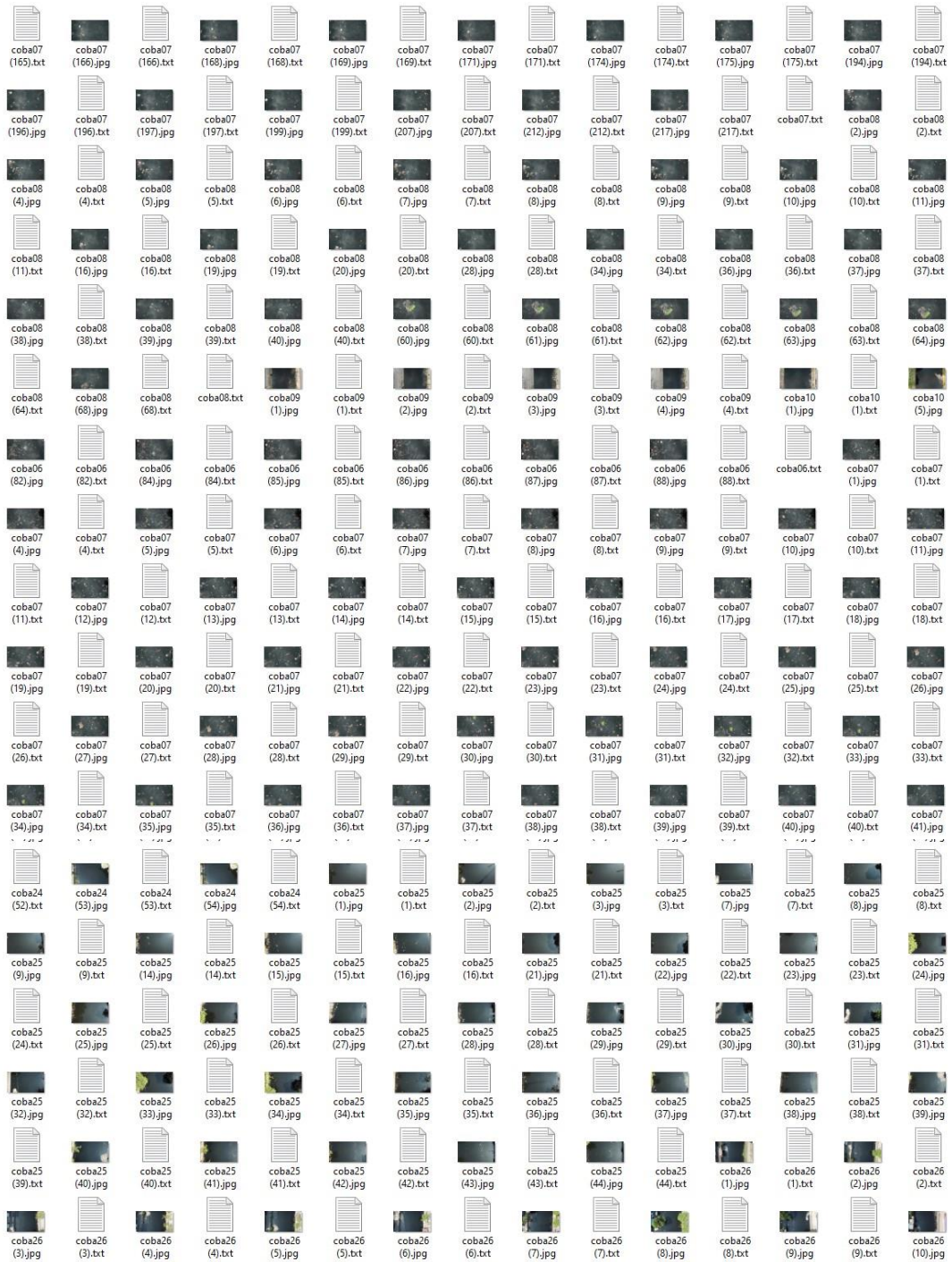
- Aprilia Safitri. 2018. "Deep Learning Dan Manfaatnya Bagi Perkembangan AI | by IYKRA | Iykra | Medium." July 31, 2018. <https://medium.com/iykra/deep-learning-dan-manfaatnya-bagi-perkembangan-ai-cab94e20c19a>.
- Billy Adytya. 2020. "11 Jenis Jenis Sampah Berdasarkan Sifat, Bentuk Dan Sumbernya | Merdeka.Com." June 18, 2020. <https://www.merdeka.com/trending/11-jenis-jenis-sampah-berdasarkan-sifat-bentuk-dan-sumbernya-klm.html?page=all>.
- Darmanto, Heri, and AMIK Taruna Probolinggo. 2019. "PENGENALAN SPESIES IKAN BERDASARKAN KONTUR OTOLITH MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK" 2: 19.
- Dinas Komunikasi dan Informatika. 2019. "Kanal Jadi Fokus Pemerintah Kota Makassar – WEBSITE RESMI PEMERINTAH KOTA MAKASSAR." June 19, 2019. <https://makassarkota.go.id/kanal-jadi-fokus-pemerintah-kota-makassar/>.
- Eka Putra, Wayan Suartika. 2016. "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101." Jurnal Teknik ITS 5 (1). <https://doi.org/10.12962/j23373539.v5i1.15696>.
- Koran Fajar. 2004. "Kanal Kita, Wajah Kita." November 21, 2004. <http://www.ampl.or.id/digilib/read/kanal-kita-wajah-kita/22161>.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2017. "ImageNet Classification with Deep Convolutional Neural Networks." Communications of the ACM 60 (6): 84–90. <https://doi.org/10.1145/3065386>.
- Marifatul Azizah, Laila, Sitti Fadillah Umayah, and Febriyana Fajar. 2018. "Deteksi Kecacatan Permukaan Buah Manggis Menggunakan Metode Deep Learning dengan Konvolusi Multilayer." Semesta Teknika 21 (2). <https://doi.org/10.18196/st.212229>.
- Muhammad Nur Abdurrahman. 2019. "Jorok! Sampah Bertebaran Di Kanal Jongaya Makassar." December 17, 2019. <https://news.detik.com/berita/d-4826252/jorok-sampah-bertebaran-di-kanal-jongaya-makassar>.
- Narkhede, Sarang. 2019. "Understanding Confusion Matrix." Medium. August 29, 2019. <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>.
- Nugraha, Yogi Adi. 2014. "Implementasi Sistem Otomatis Pada Robot Kapal Berbasis Komputer Vision Untuk Kontes Kapal Cepat Tak Berawak Nasional (KKCTBN)."
- Panella, F., J. Boehm, Y. Loo, A. Kaushik, and D. Gonzalez. 2018. "DEEP LEARNING AND IMAGE PROCESSING FOR AUTOMATED CRACK DETECTION AND DEFECT MEASUREMENT IN UNDERGROUND STRUCTURES." ISPRS - International Archives of the Photogrammetry,

- Remote Sensing and Spatial Information Sciences XLII–2 (May): 829–35.  
<https://doi.org/10.5194/isprs-archives-XLII-2-829-2018>.
- Radhif, Azwar. 2019. “Permasalahan Kanal Kota Makassar, Mengalihkan Fungsi Kanal Yang Sebenarnya.” November 20, 2019.  
<https://maupa.co/permasalahan-kanal-kota-makassar-mengalihkan-fungsi-kanal-yang-sebenarnya/>.
- Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2015. “You Only Look Once: Unified, Real-Time Object Detection.” ArXiv:1506.02640 [Cs], June. <http://arxiv.org/abs/1506.02640>.
- Rosebrock, Adrian. 2016. “Intersection over Union (IoU) for Object Detection.” PyImageSearch (blog). November 7, 2016.  
<https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>.
- Sharma, Atharva, Xiuwen Liu, Xiaojun Yang, and Di Shi. 2017. “A Patch-Based Convolutional Neural Network for Remote Sensing Image Classification.” *Neural Networks* 95 (November): 19–28.  
<https://doi.org/10.1016/j.neunet.2017.07.017>.
- Socher, Richard, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. n.d. “Convolutional-Recursive Deep Learning for 3D Object Classification,” 9.
- Szeliski, Richard. 2010. *Computer Vision - Algorithms and Applications*.
- Wahyuni, Tri. 2016. “Indonesia Penyumbang Sampah Plastik Terbesar Ke-Dua Dunia.” *Gaya Hidup*. 2016. <https://www.cnnindonesia.com/gaya-hidup/20160222182308-277-112685/indonesia-penyumbang-sampah-plastik-terbesar-ke-dua-dunia>.
- WD, D. 2018. “Soal Plastik Di Laut, Indonesia Terancam Digugat Di Mahkamah Internasional | SOSBUD: Laporan Seputar Seni, Gaya Hidup Dan Sosial | DW | 21.02.2018.” 2018. <https://www.dw.com/id/soal-plastik-di-laut-indonesia-terancam-digugat-di-mahkamah-internasional/a-42677575>.
- Xin, Yang, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang. 2018. “Machine Learning and Deep Learning Methods for Cybersecurity.” *IEEE Access* 6: 35365–81.  
<https://doi.org/10.1109/ACCESS.2018.2836950>.
- Zhang, Lu, Jianjun Tan, Dan Han, and Hao Zhu. 2017. “From Machine Learning to Deep Learning: Progress in Machine Intelligence for Rational Drug Discovery.” *Drug Discovery Today* 22 (11): 1680–85.  
<https://doi.org/10.1016/j.drudis.2017.08.010>.
- Zhang, Xiangyu, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.” In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6848–56. Salt Lake City, UT: IEEE.  
<https://doi.org/10.1109/CVPR.2018.00716>.

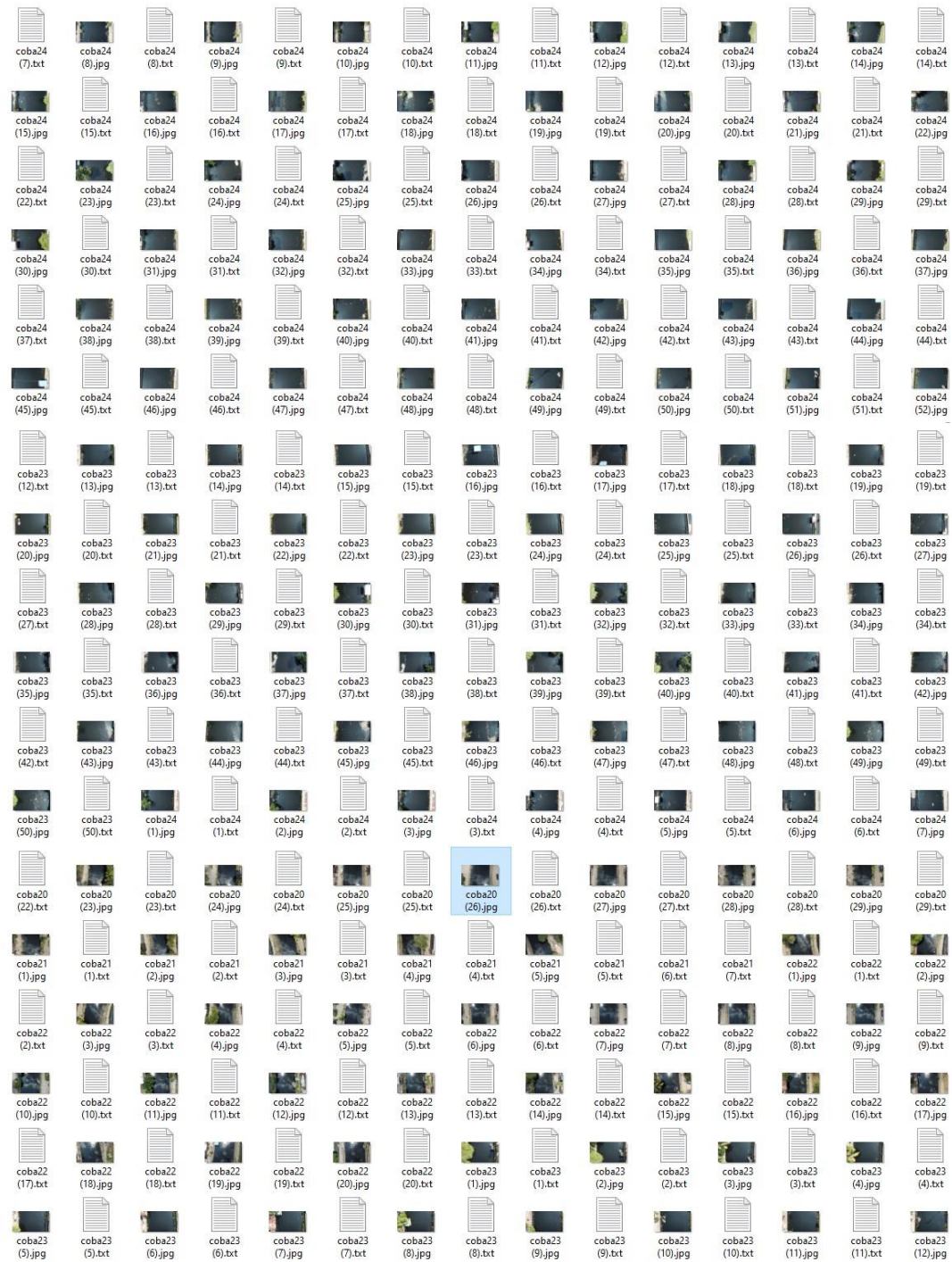
# LAMPIRAN

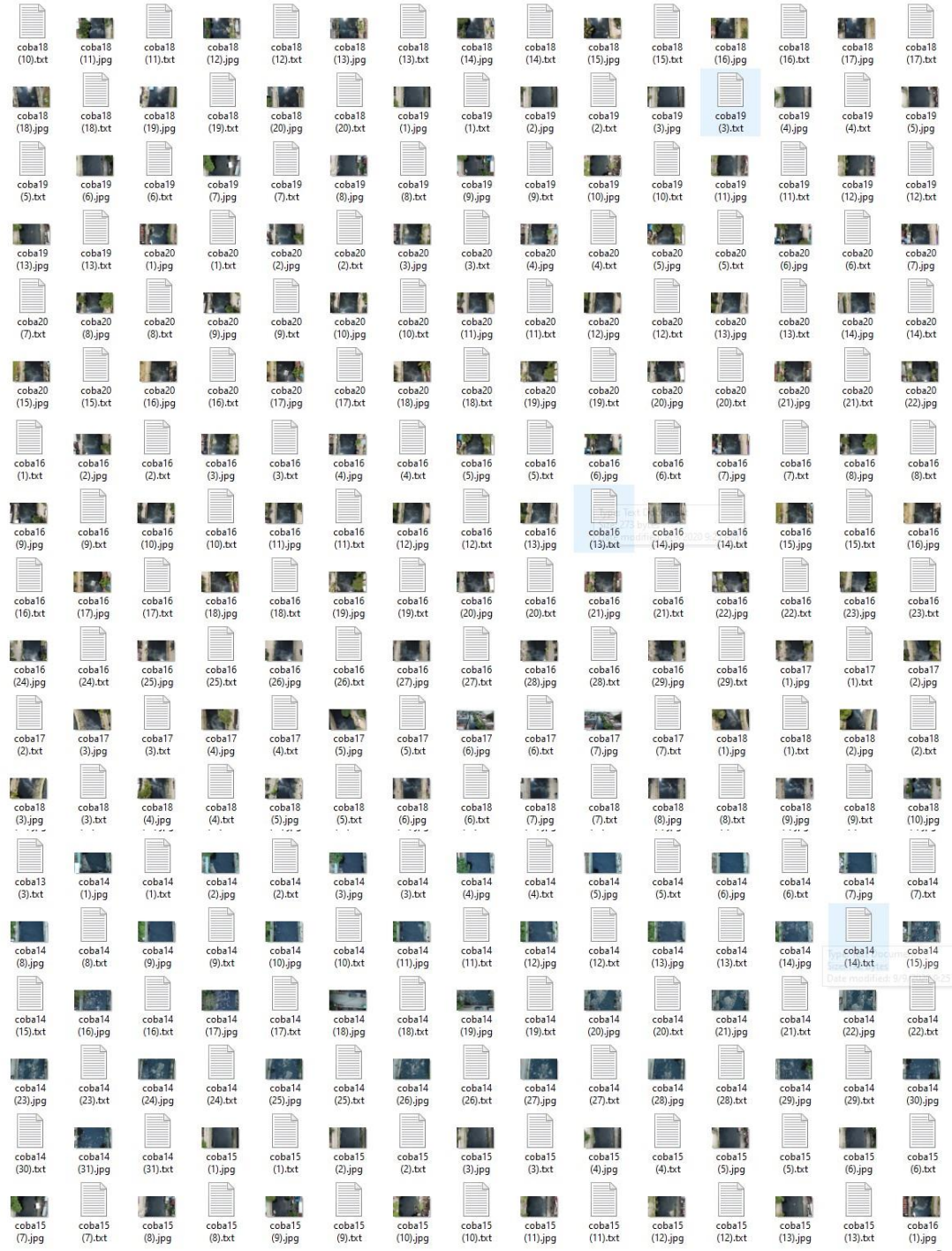
## Lampiran 1. Data *Training* Sistem

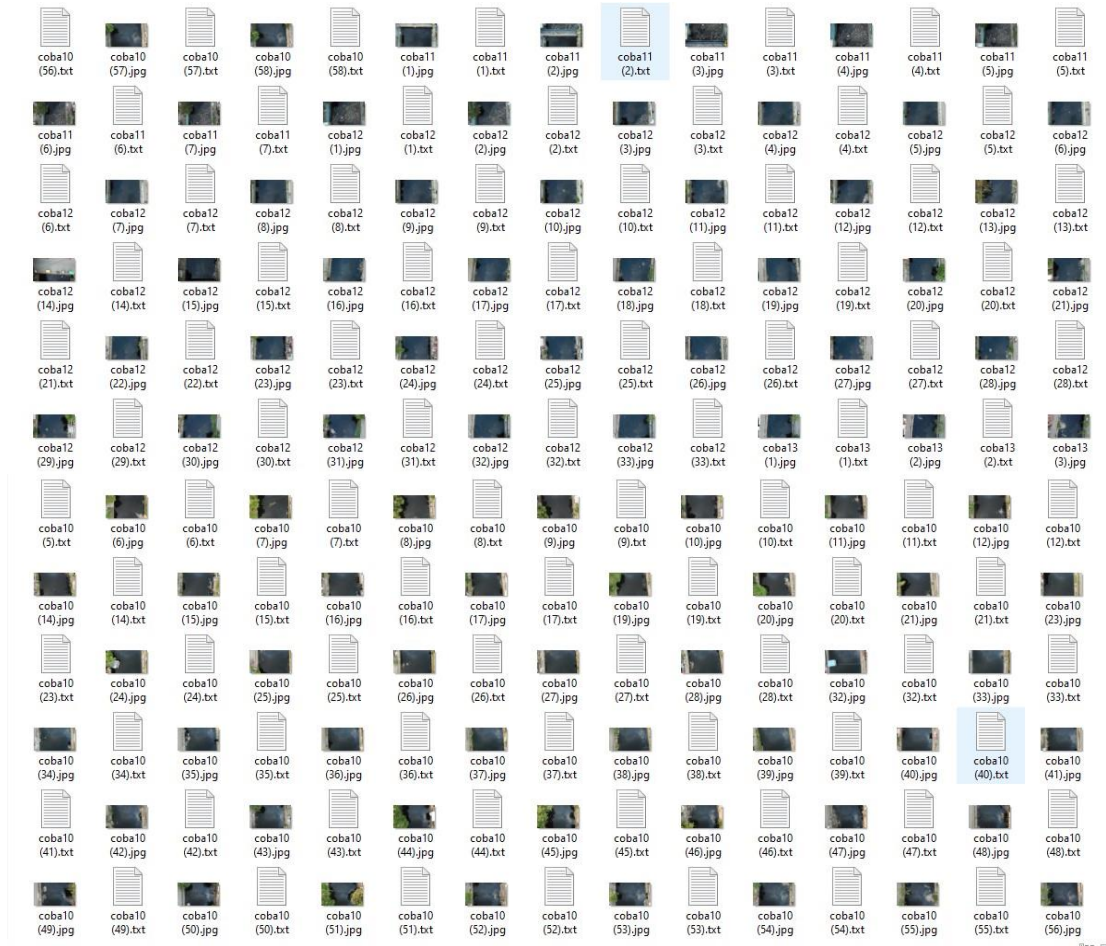










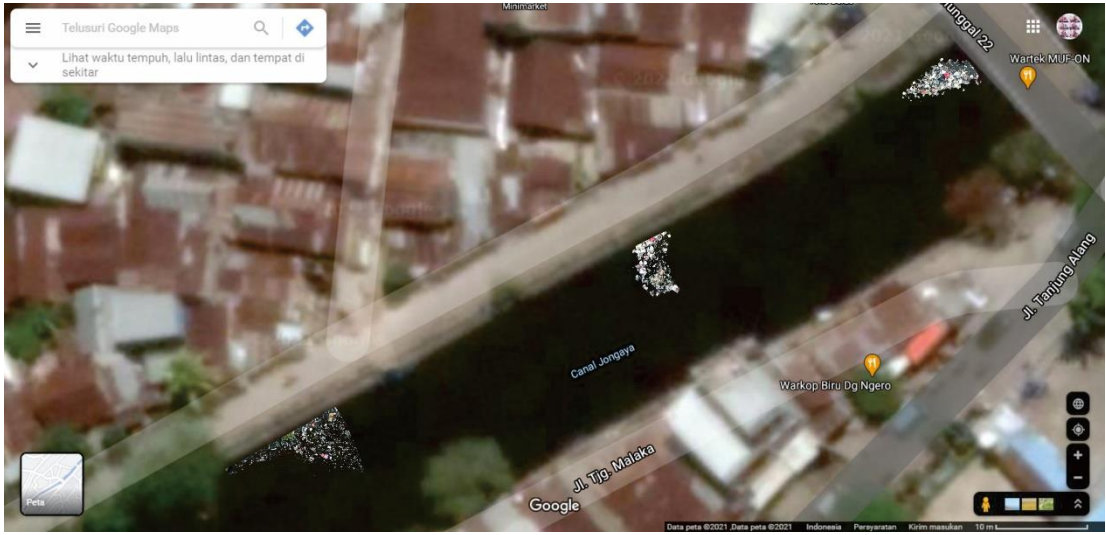
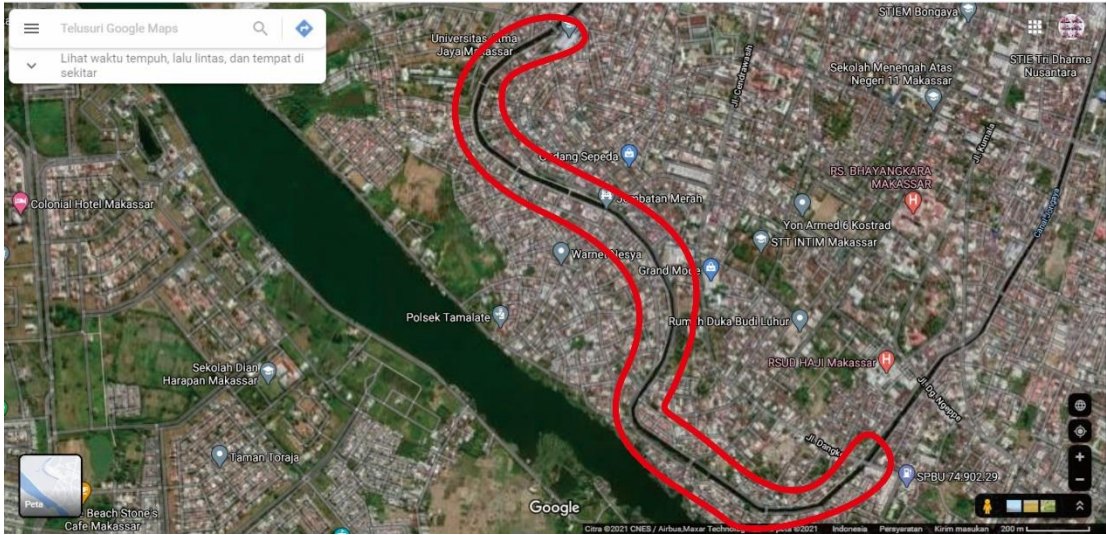


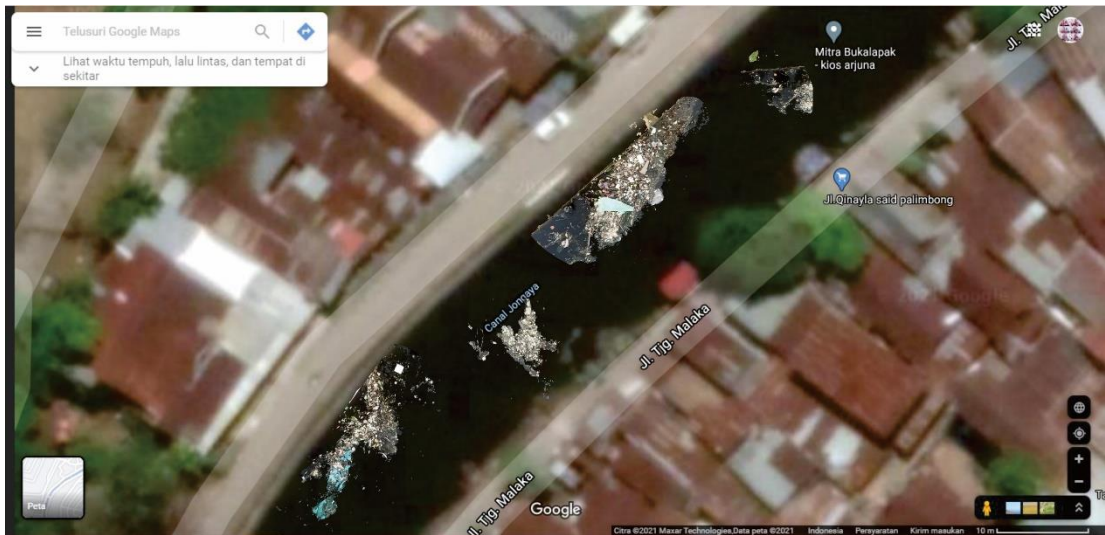




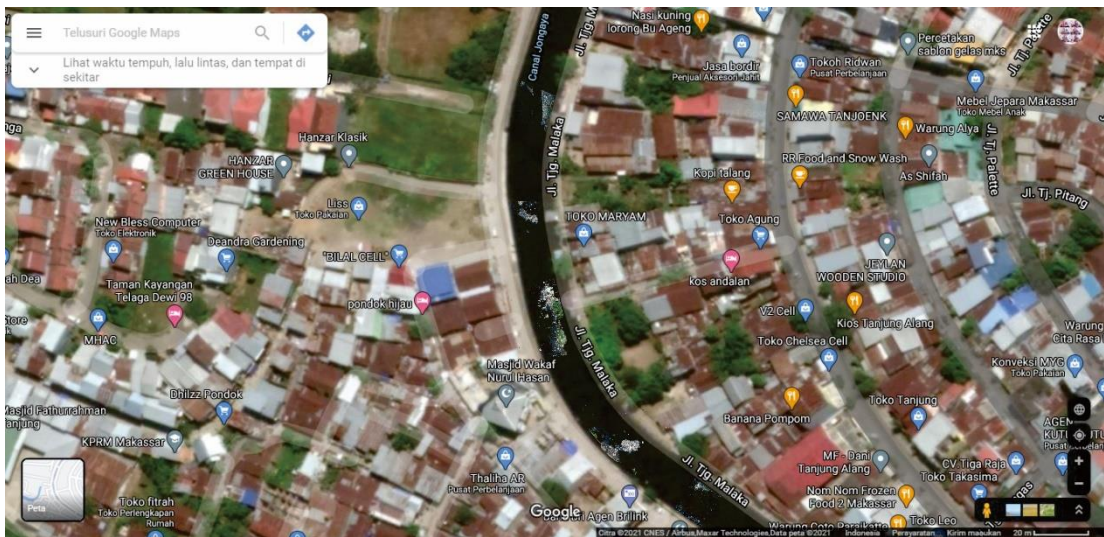
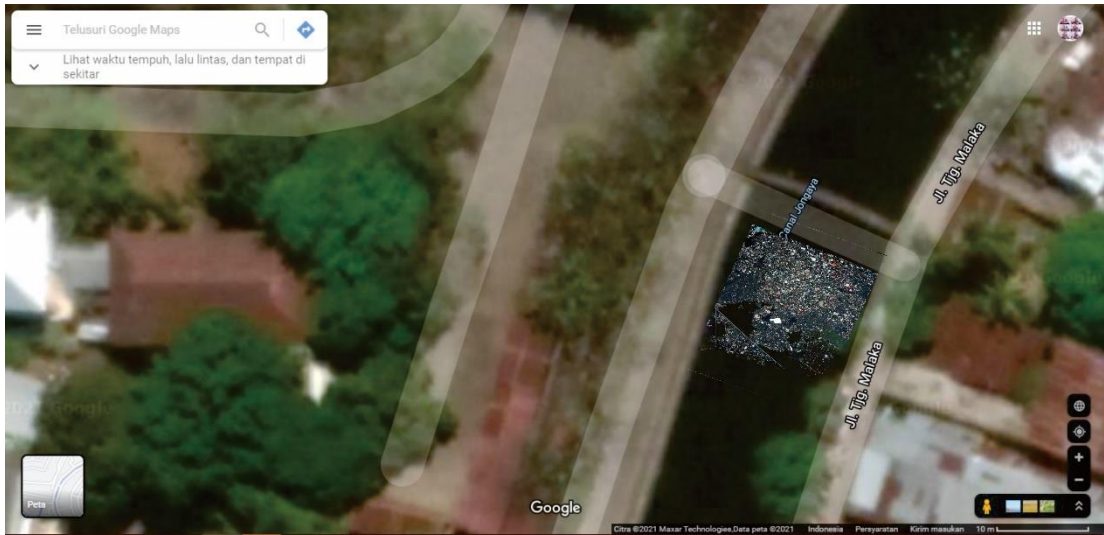


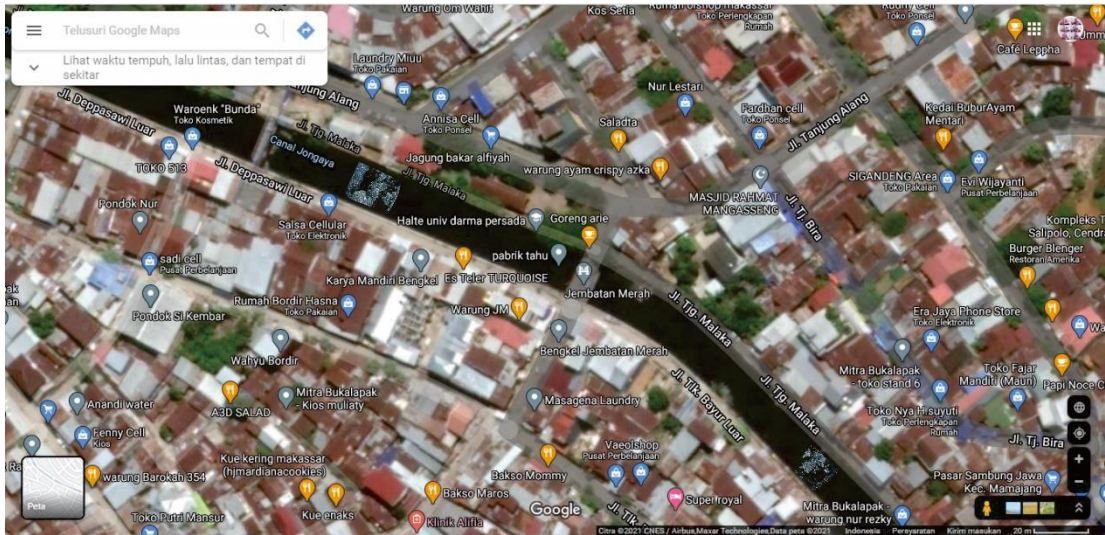
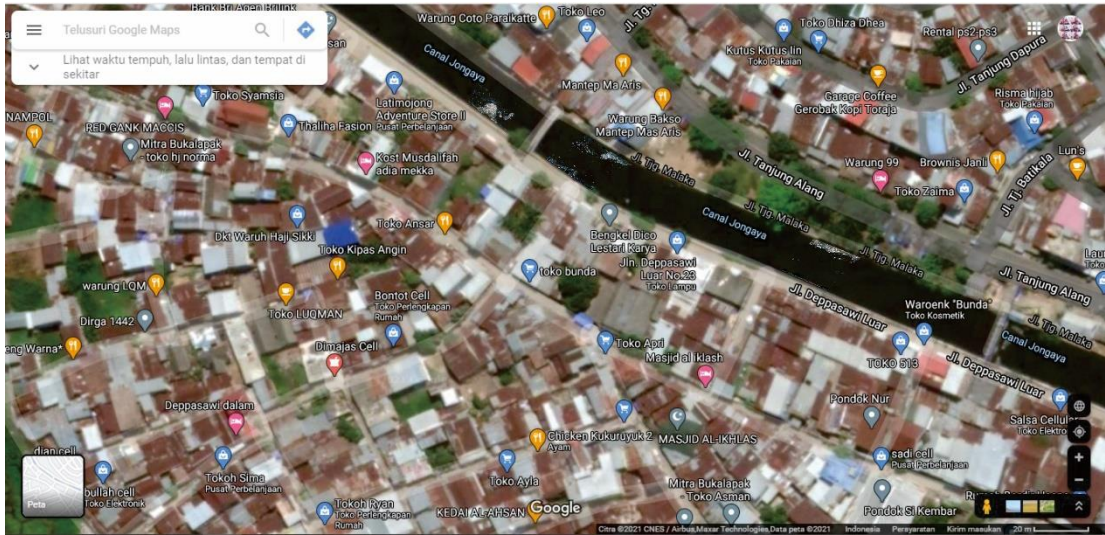
**Lampiran 2.** Gambar aliran kanal untuk mengetahui letak tempat pengambilan data uji sistem



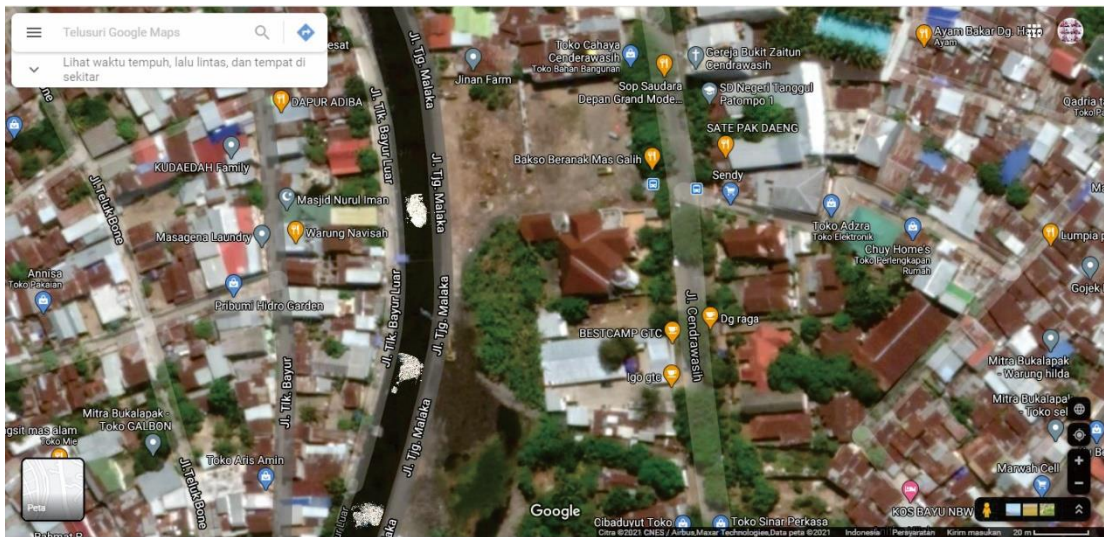
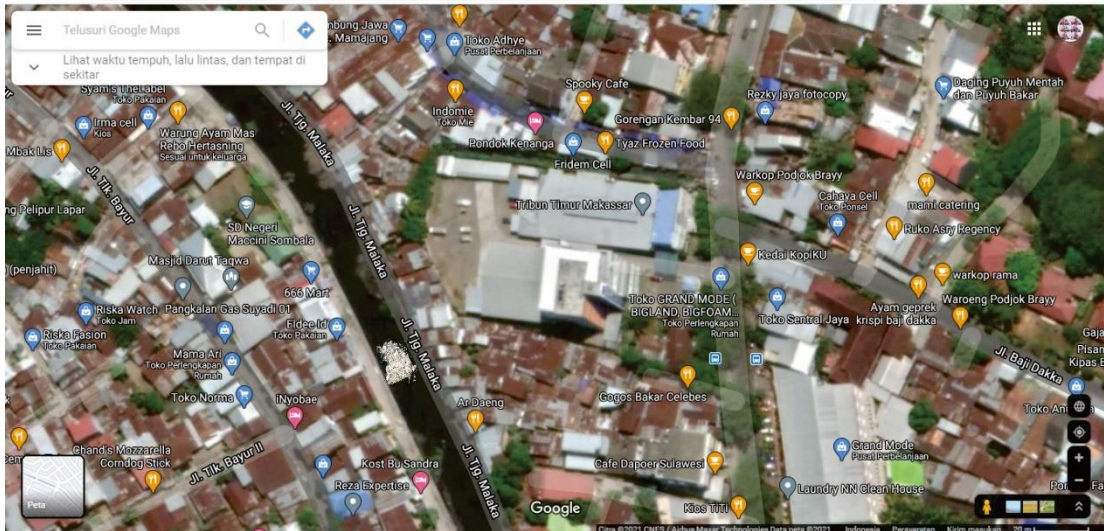








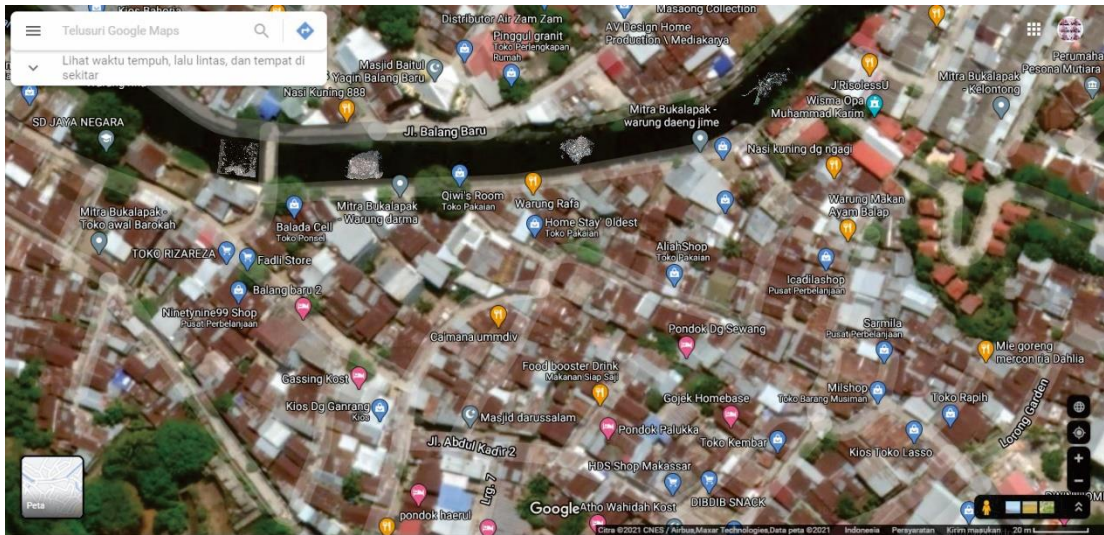
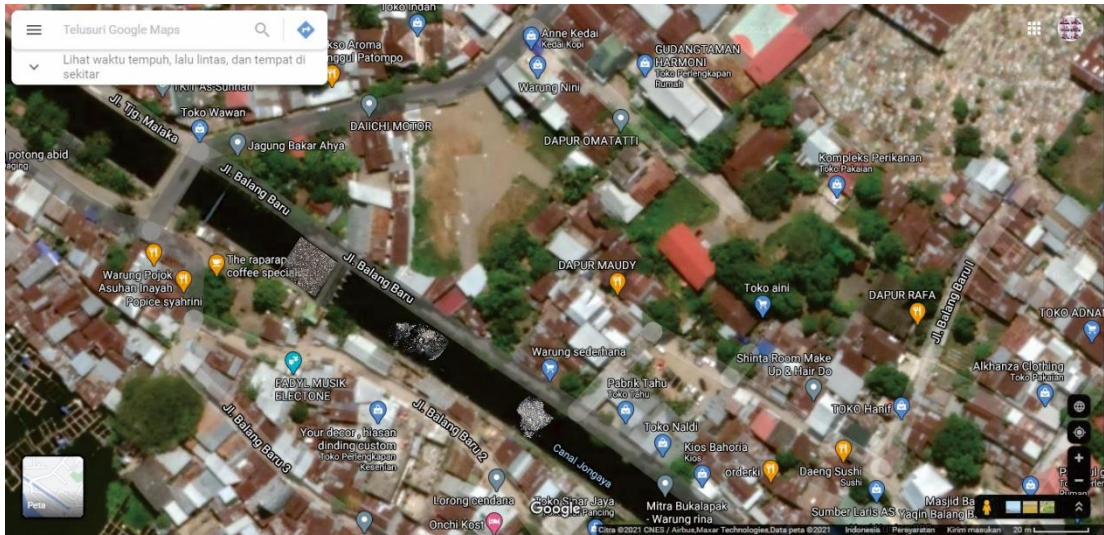


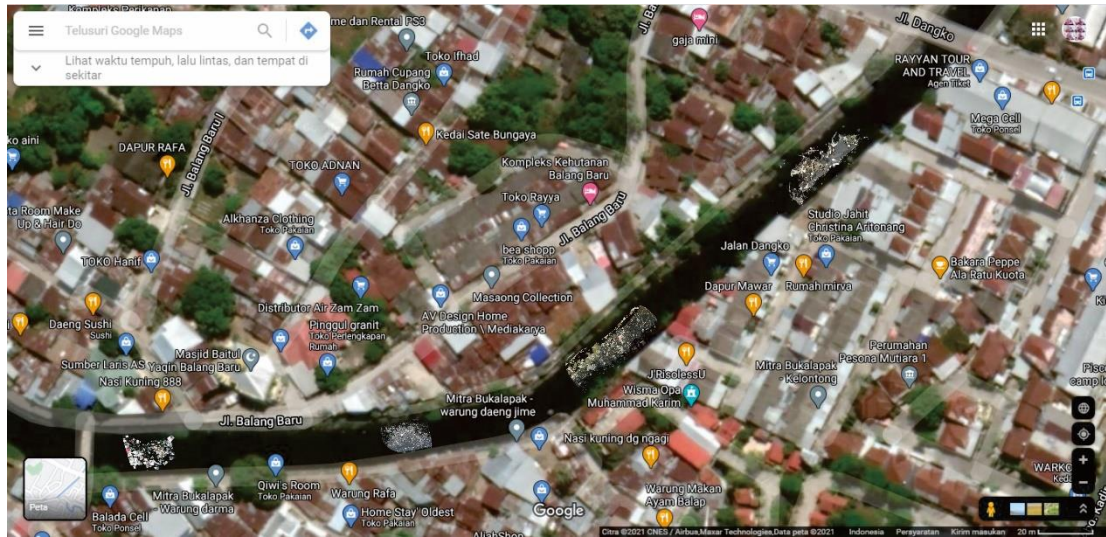














### Lampiran 3. File Konfigurasi *Training* YOLOv3.cfg

```
1 [net]
2 # Testing
3 #batch = 1
4 #subdivisions = 1
5 # Training
6 batch = 64
7 subdivisions = 16
8 width = 416
9 height = 416
10 channels = 3
11 momentum = 0.9
12 decay = 0.0005
13 angle = 0
14 saturation = 1.5
15 exposure = 1.5
16 hue = .1
17
18 learning_rate = 0.001
19 burn_in = 1000
20 max_batches = 4000
21 policy = steps
22 steps = 3200,3600
23 scales = .1,.1
24
25 [convolutional]
26 batch_normalize=1
27 filters = 32
28 size = 3
29 stride = 1
30 pad = 1
31 activation = leaky
```

```
764 [convolutional]
765 batch_normalize=1
766 size=3
767 stride=1
768 pad=1
769 filters=256
770 activation=leaky
771
772 [convolutional]
773 size=1
774 stride=1
775 pad=1
776 filters=21
777 activation=linear
778
779
780 [yolo]
781 mask = 0,1,2
782 anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326
783 classes=2
784 num=9
785 jitter=.3
786 ignore_thresh = .7
787 truth_thresh = 1
788 random=1
789
```

#### Lampiran 4. Kode Program Pengujian Sistem

```
# import packages
import numpy as np
import argparse
import time
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="path to input image")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
                help="minimum probability to filter weak detections")
ap.add_argument("-t", "--threshold", type=float, default=0.5,
                help="threshold when applying non-maxima suppression")
args = vars(ap.parse_args())

# load the YOLO class labels
labelsPath = "data/obj.names"
LABELS = open(labelsPath).read().strip().split("\n")

# paths to the YOLO weights and model configuration
weightsPath = "data/sampah_final.weights"
configPath = "data/sampah.cfg"

# initialize a list of colors to represent each possible class label
COLORS = (103, 220, 225)
# np.random.seed(42)
# COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
#                               dtype="uint8")
```

```

# load our YOLO object detector
print("Processing...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# load input image and grab its spatial dimensions
image = cv2.imread(args["image"])
# image = cv2.resize(image, (0,0), fx=0.7, fy=0.7)
(H, W) = image.shape[:2]

# determine only the *output* layer names that we need from YOLO
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# construct a blob from the input image and then perform a forward
# pass of the YOLO object detector, giving us our bounding boxes and
# associated probabilities
blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True, crop=False)
net.setInput(blob)
start = time.time()
layerOutputs = net.forward(ln)
end = time.time()
print("Nilai blob: {}".format(blob.shape))

# initialize our lists of detected bounding boxes, confidences, and
# class IDs, respectively
boxes = []
confidences = []
classIDs = []

# loop over each of the layer outputs

```

for output in layerOutputs:

```
# loop over each of the detections
```

```
for detection in output:
```

```
# extract the class ID and confidence (i.e., probability) form Image
```

```
scores = detection[5:]
```

```
classID = np.argmax(scores)
```

```
confidence = scores[classID]
```

```
# print ("Nilai Confidence dari object ialah :", confidence)
```

```
# print ("Nilai ID dari Class ialah :", classID)
```

```
# filter out weak predictions by ensuring the detected
```

```
# probability is greater than the minimum probability
```

```
if confidence > args["confidence"]:
```

```
# scale the bounding box coordinates back relative to the size of the
```

image

```
box = detection[0:4] * np.array([W, H, W, H])
```

```
(centerX, centerY, width, height) = box.astype("int")
```

```
# print ("Nilai dari B.Box ialah :", centerX, " ", centerY, " ", width, "
```

", height)

```
# use the center (x, y)-coordinates to get the top and and left
```

```
#corner of the bounding box
```

```
x = int(centerX - (width / 2))
```

```
y = int(centerY - (height / 2))
```

```
print ("Nilai x dan y dari B.Box ialah :", x, " ", y)
```

```
# update our list of bounding box coordinates, confidences, and class
```

IDs

```
boxes.append([x, y, int(width), int(height)])
```

```
confidences.append(float(confidence))
```

```
classIDs.append(classID)
```



```

# apply non-maxima suppression to suppress weak, overlapping bounding boxes
idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"], args["threshold"])

# ensure at least one detection exists
if len(idxs) > 0:
    # loop over the indexes
    for i in idxs.flatten():
        # extract the bounding box coordinates
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])

        # draw a bounding box rectangle and label on the image
        # color = [int(c) for c in COLORS[classIDs[i]]]
        cv2.rectangle(image, (x, y), (x + w, y + h), COLORS, 8)
        text = "{}: {:.4f}".format(LABELS[classIDs[i]], confidences[i])
        print(text)
        cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 2,
COLORS, 8)
        # cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.2,
COLORS, 2)

# Font type
font = cv2.FONT_HERSHEY_SIMPLEX
# Font Coordinate
org = (40, 480)
# Font Size
fontScale = 10
# Font color with format (B,G,R)
color = (206, 0, 0)
# Font Thickness

```

```
thickness = 20
```

```
# show timing information on YOLO
```

```
print("YOLO took {:.6f} seconds".format(end - start))
```

```
# Save the output image
```

```
cv2.waitKey(0)
```