# DAFTAR PUSTAKA

A, S., B V, D., Department of CSE Sri sairam College of engineering Anekal, S C, N., & N S, A. (2023). Image Processing. *International Journal of Innovative Research in Information Security*, *09*(03), 79–83. https://doi.org/10.26562/ijiris.2023.v0903.06

Amin, U., Shahzad, M. I., Shahzad, A., Shahzad, M., Khan, U., & Mahmood, Z. (2023). Automatic Fruits Freshness Classification Using CNN and Transfer Learning. *Applied Sciences*, *13*(14), 8087. https://doi.org/10.3390/app13148087

Bacchi, S., Oakden-Rayner, L., Zerner, T., Kleinig, T., Patel, S., & Jannes, J. (2019). Deep Learning Natural Language Processing Successfully Predicts the Cerebrovascular Cause of Transient Ischemic Attack-Like Presentations. *Stroke*, *50*(3), 758–760. https://doi.org/10.1161/STROKEAHA.118.024124

Basri, H., Syarif, I., Sukaridhoto, S., & Falah, M. F. (2019). INTELLIGENT SYSTEM FOR AUTOMATIC CLASSIFICATION OF FRUIT *DEFECT* USING FASTER REGION-BASED CONVOLUTIONAL NEURAL NETWORK (FASTER R-CNN). *Kursor*, *10*(1). https://doi.org/10.28961/kursor.v10i1.187

Daryono, B., & Tammu, R. (2022). *Karakteristik, Potensi Genetik, dan Pemanfaatan Cabai Katokkon Asal Toraja, Indonesia*.

Dhiman, P., Kaur, A., Hamid, Y., Alabdulkreem, E., Elmannai, H., & Ababneh, N. (2023). Smart Disease Detection System for Citrus Fruits Using Deep

Learning with Edge Computing. *Sustainability*, *15*(5), 4576. https://doi.org/10.3390/su15054576

Duranova, H., Valkova, V., & Gabriny, L. (2022). Chili peppers (Capsicum spp.): The spice not only for cuisine purposes: an update on current knowledge. *Phytochemistry Reviews*, *21*(4), 1379–1413. https://doi.org/10.1007/s11101-021-09789-7

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Gupta, S. (2022). Computer Vision. In R. Gupta, A. K. Rana, S. Dhawan, & K. Cengiz, *Advanced Sensing in Image Processing and IoT* (1st ed., pp. 19–42). CRC Press. https://doi.org/10.1201/9781003221333-2

IBM. *What is Deep Learning? | IBM*. Retrieved September 29, 2023, from https://www.ibm.com/topics/deep-learning

Khatun, M., Ali, F., Nine, J., & Sarker, P. (2020). Fruits Classification using Convolutional Neural Network. *ResearchGate*, *5*(8), 6.

Kumar, M. S. (2023). PATTERN RECOGNITION FROM IMAGES AND VIDEOS USING MACHINE LEARNING. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, *07*(02). https://doi.org/10.55041/IJSREM17690

Li, H., Ota, K., & Dong, M. (2018). Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Network*, *32*(1), 96–101. https://doi.org/10.1109/MNET.2018.1700202

Liu, Z., Xiang, X., Qin, J., YunTan, Zhang, Q., & N. Xiong, N. (2020). Image Recognition of Citrus Diseases Based on Deep Learning. *Computers,*

*Materials & Continua*, *66*(1), 457–466. https://doi.org/10.32604/cmc.2020.012165

Manaswi, N. K. (2018). Convolutional Neural Networks. In N. K. Manaswi, *Deep Learning with Applications Using Python* (pp. 91–96). Apress. https://doi.org/10.1007/978-1-4842-3516-4_6

Mimma, N.-E.-A., Ahmed, S., Rahman, T., & Khan, R. (2022). Fruits Classification and Detection Application Using Deep Learning. *Scientific Programming*, *2022*, 1–16. https://doi.org/10.1155/2022/4194874

Namdev, U., Agrawal, S., & Pandey, R. (2022). Object Detection Techniques based on Deep Learning: A Review. *Computer Science & Engineering: An International Journal*, *12*(1), 125–134. https://doi.org/10.5121/cseij.2022.12113

PVTPP 2014. *Berita Resmi PVT Pendaftaran Varietas Lokal No. Publikasi: 055/BR/PVL/02/2014*. http://pvtpp.setjen.pertanian.go.id/berita-resmi/pendaftaran-varietas-lokal/padi-nama-varietas-katokkon/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection* (arXiv:1506.02640). arXiv. http://arxiv.org/abs/1506.02640

Satyanarayanan, M. (2017). Edge Computing. *Computer*, *50*(10), 36–38. https://doi.org/10.1109/MC.2017.3641639

Tian, Y. (2020). Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm. *IEEE Access*, *8*, 125731–125744. https://doi.org/10.1109/ACCESS.2020.3006097

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors* (arXiv:2207.02696). arXiv. http://arxiv.org/abs/2207.02696

Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information* (arXiv:2402.13616). arXiv. http://arxiv.org/abs/2402.13616

# LAMPIRAN

Lampiran 1 Contoh Dataset Cabai Katokkon

Grade A

Grade B

Defect

Fly Bites

# Lampiran 2 *Source Code* Training Model YOLOv8 dan YOLOv9

## Install Ultralytics

```
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/MyDrive/Colab Notebooks/YOLOv8


# Ultralytics version for MODEL CONVERSION
# !git clone https://github.com/ultralytics/ultralytics
%pip install -qe ultralytics
%cd ultralytics


# MODEL TRAINING
%pip install ultralytics
```

## Import Dataset from Roboflow

```
#YOLOv8 & YOLOv9 dataset used the same format
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="YKs5Ocm1gTqydIgP567E")
project = rf.workspace("unhas-9yowg").project("katokkon_ab_defect_flybites")
version = project.version(4)
dataset = version.download("yolov8")
```

## Train YOLOv8

```
# hyperparameter tuning

from ultralytics import YOLO
model = YOLO('yolov8n.pt')

model.tune(data='/content/katokkon_ab_defect_flybites-4/data.yaml', epochs=50, iterations=25, optimizer='AdamW', plots=False, save=Fals

# yolov8
!yolo task=detect \
      mode=train \
      model=yolov8n.pt \
      data='/content/katokkon_ab_defect_flybites-4/data.yaml' \
      epochs= 100\
      imgsz=320
```

## Train YOLOv9

```
# yolov9
!yolo task=detect \
      mode=train \
      model=yolov9c.pt \
      data='/content/katokkon_ab_defect_flybites-4/data.yaml' \
      epochs= 300\
      patience=20\
      imgsz=320
```

## Tensorboard

```
%load_ext tensorboard
%tensorboard --logdir runs
```

## Export model to ONNX format

```
!yolo task=detect mode=export model="/content/drive/MyDrive/Colab Notebooks/YOLOv8/runs/detect/train7-first100epochwnewdatasetlabelw-ba
```

## export to ncnn & tflite

```
!yolo task=detect mode=export model="/content/drive/MyDrive/Colab Notebooks/YOLOv8/train4.pt" format=tflite simplify=True opset=13 imgs

!yolo task=detect mode=export model="/content/drive/MyDrive/Colab Notebooks/YOLOv8/v9train2.pt" format=tflite simplify=True opset=13 im
```

# Lampiran 3 *Source Code* Training Model YOLOv7

## Install Dependencies

```
!python --version


# Download YOLOv7 repository and install requirements
!git clone https://github.com/WongKinYiu/yolov7
%cd yolov7
!pip install -r requirements.txt
```

## Download dataset

```
# chili dataset
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="YKs5Ocm1gTqydIgP567E")
project = rf.workspace("unhas-9yowg").project("katokkon_ab_defect_flybites")
version = project.version(4)
dataset = version.download("yolov7")
```

## Training YOLOv7

```
# download COCO starting checkpoint
%cd /content/yolov7
!wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7_training.pt
```

## generate confusion matrix yolov7: edit utils/metrics.py -- line 168

```
# run this cell to begin training
%cd /content/yolov7
!python train.py --batch 16 --epochs 50 --data {dataset.location}/data.yaml --weights 'yolov7_training.pt' --device 0
```

```
                 from  n    params  module                                  arguments
  0                -1  1       928  models.common.Conv                      [3, 32, 3, 1]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1      8320  models.common.Conv                      [128, 64, 1, 1]
  5                -2  1      8320  models.common.Conv                      [128, 64, 1, 1]
  6                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  7                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  8                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  9                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
 10  [-1, -3, -5, -6]  1         0  models.common.Concat                    [1]
 11                -1  1     66048  models.common.Conv                      [256, 256, 1, 1]
 12                -1  1         0  models.common.MP                        []
 13                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 14                -3  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 16        [-1, -3]  1         0  models.common.Concat                    [1]
 17                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 18                -2  1     33024  models.common.Conv                      [256, 128, 1, 1]
 19                -1  1    147712  models.common.Conv                      [128, 128, 3, 1]
 20                -1  1    147712  models.common.Conv                      [128, 128, 3, 1]
```

```
35                -1  1    590336  models.common.Conv                      [256, 256, 3, 1]
36  [-1, -3, -5, -6]  1         0  models.common.Concat                    [1]
37                -1  1   1050624  models.common.Conv                      [1024, 1024, 1, 1]
38                -1  1         0  models.common.MP                        []
39                -1  1    525312  models.common.Conv                      [1024, 512, 1, 1]
40                -3  1    525312  models.common.Conv                      [1024, 512, 1, 1]
41                -1  1   2360320  models.common.Conv                      [512, 512, 3, 2]
42          [-1, -3]  1         0  models.common.Concat                    [1]
43                -1  1    262656  models.common.Conv                      [1024, 256, 1, 1]
44                -2  1    262656  models.common.Conv                      [1024, 256, 1, 1]
45                -1  1    590336  models.common.Conv                      [256, 256, 3, 1]
46                -1  1    590336  models.common.Conv                      [256, 256, 3, 1]
47                -1  1    590336  models.common.Conv                      [256, 256, 3, 1]
48                -1  1    590336  models.common.Conv                      [256, 256, 3, 1]
49  [-1, -3, -5, -6]  1         0  models.common.Concat                    [1]
50                -1  1   1050624  models.common.Conv                      [1024, 1024, 1, 1]
51                -1  1   7609344  models.common.SPPCSPC                   [1024, 512, 1]
52                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
53                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
54                37  1    262656  models.common.Conv                      [1024, 256, 1, 1]
```

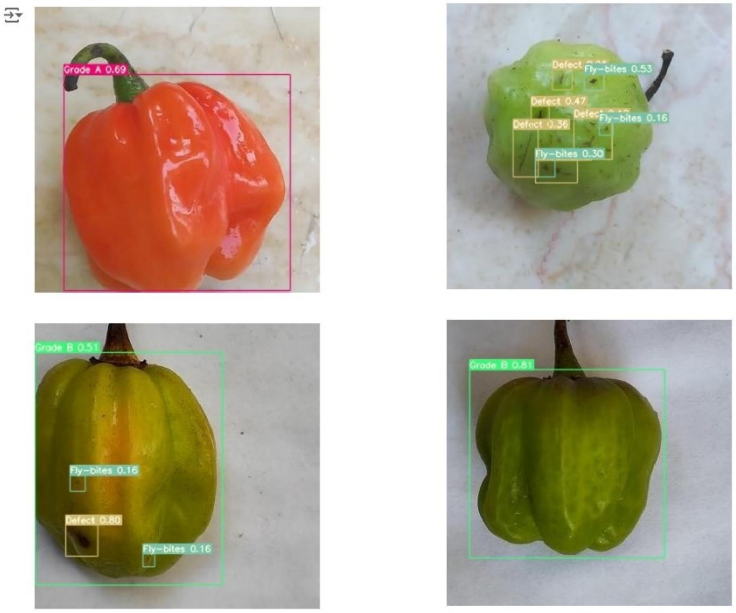
## Evaluation

```
# Run evaluation
!python detect.py --weights runs/train/exp/weights/best.pt --conf 0.1 --source {dataset.location}/test/images
```

```
#display inference on ALL test images

import glob
from IPython.display import Image, display

i = 0
limit = 10000 # max images to print
for imageName in glob.glob('/content/yolov7/runs/detect/exp/*.jpg'): #assuming JPG
    if i < limit:
      display(Image(filename=imageName))
      print("\n")
    i = i + 1
```



Lampiran 4 *Source Code* Inferensi *Raspberry Pi*

```
import cv2
import argparse
import time
import serial


from ultralytics import YOLO
import supervision as sv

def parse_arguments() -> argparse.Namespace:
parser = argparse.ArgumentParser(description="YOLOv8 Live")
parser.add_argument(
"--webcam-resolution",
default=[1280, 720],
nargs=2,
type=int
)
parser.add_argument(
"--modeldir",
default="yolov8n.pt",
help='Folder the .pt file is located in'
)
parser.add_argument(
"--source",
default=0,
type=int
)
args = parser.parse_args()
return args

def write_list_to_file(my_list, filename):
"""
Writes a list to a text file with each element on a new line.

Args:
my_list: The list to be written to the file.
filename: The name of the text file.
"""
with open(filename, 'w') as f:
for item in my_list:
f.seek(0, 2)
f.write(str(item) + '\n') # Convert each item to string and add newline


results_list = []

def main():
args = parse_arguments()
frame_width, frame_height = args.webcam_resolution
MODEL_NAME = args.modeldir
```

```
CAMERA_SOURCE = args.source

cap = cv2.VideoCapture(CAMERA_SOURCE)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, frame_width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, frame_height)

model = YOLO(MODEL_NAME)

box_annotator = sv.BoxAnnotator(
thickness=2,
text_thickness=2,
text_scale=1
)

#calculating fps
prev_frame_time = 0
new_frame_time = 0

#loop through image from frame
while True:

ret, frame = cap.read()

result = model(frame, agnostic_nms=True)
object_classes = result[0].boxes.cls.to('cpu').tolist()
speed_dict = result[0].speed['inference']
results_list.append(speed_dict)

#print out detection results
for i in object_classes:
print(i)


detections = sv.Detections.from_ultralytics(result[0])

labels = [
f"{model.model.names[class_id]} {confidence:.2f}"
for class_id, confidence
in zip(detections.class_id, detections.confidence)
]

frame = box_annotator.annotate(
scene=frame,
detections=detections,
labels=labels
)

font = cv2.FONT_HERSHEY_SIMPLEX
new_frame_time = time.time()

#calculating fps
fps = 1/(new_frame_time-prev_frame_time)
prev_frame_time = new_frame_time
fps = float("{:.2f}".format(fps))
fps = str(fps)
```

```
cv2.putText(frame,fps,(7,35), font, 1, (100, 255, 0), 1, cv2.LINE_AA)

cv2.imshow("yolo", frame)

if (cv2.waitKey(30) == 27):
break

#write_list_to_file(results_list, "yolov8-inference.txt")

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

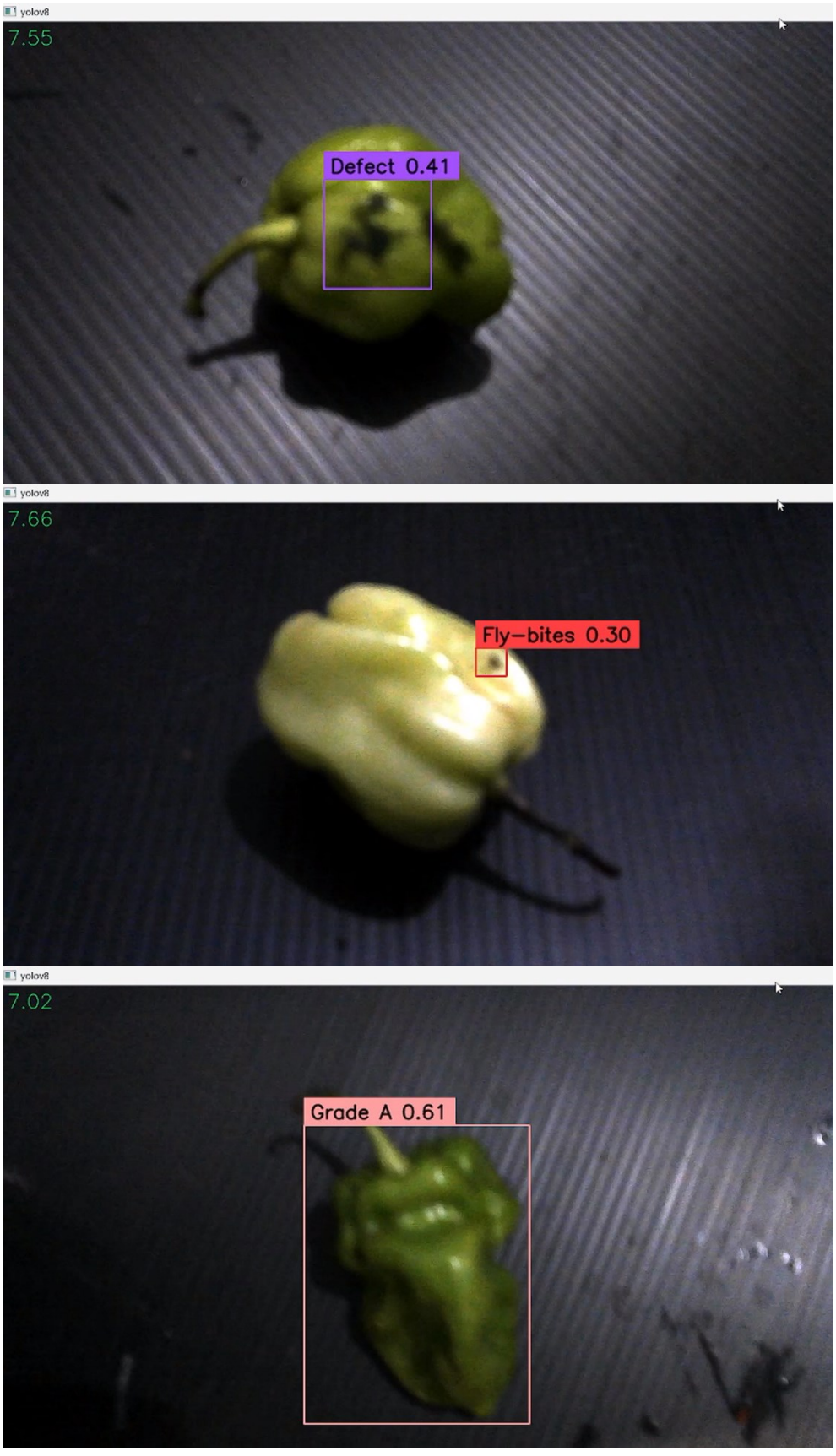
Lampiran 5 *Source Code* Inferensi *Jetson Nano*

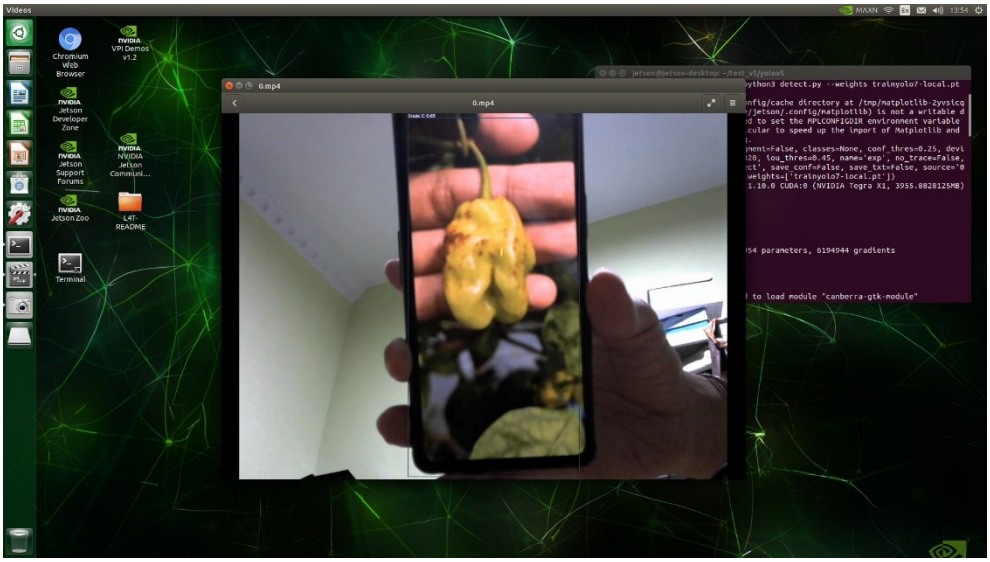
*Source Code* inferensi YOLOv7 pada *Jetson Nano*:
https://github.com/williamadamm/Katokkon-Classification/tree/master/programs/Jetson-nano

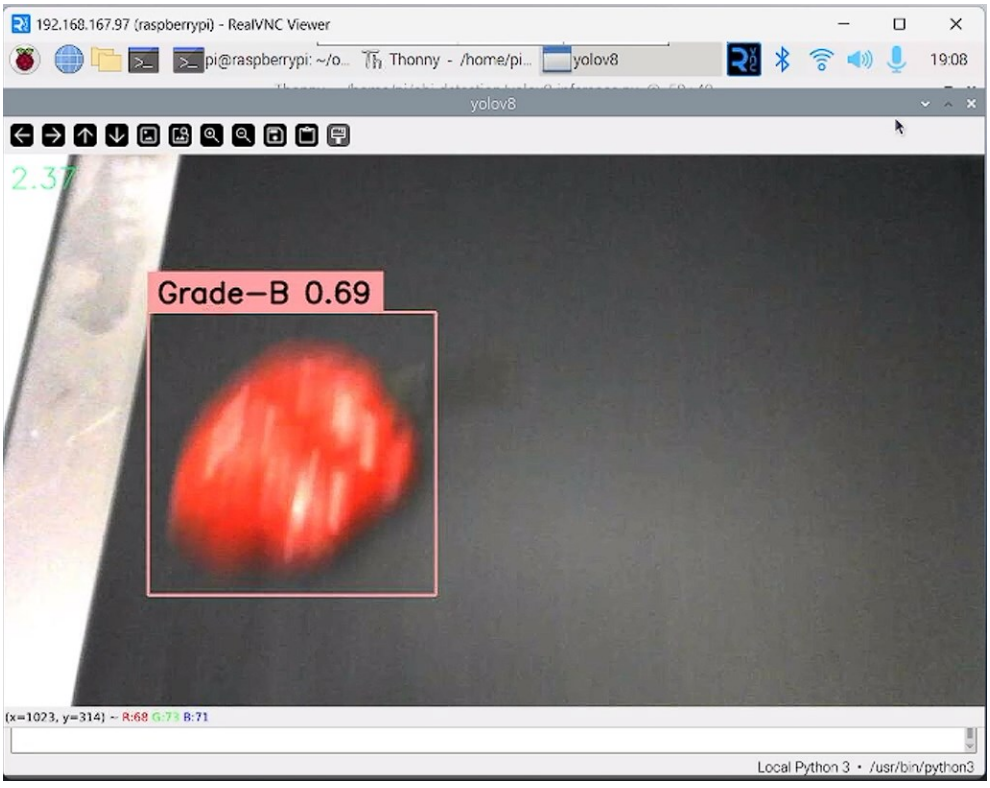
Lampiran 6 Program Aplikasi Android

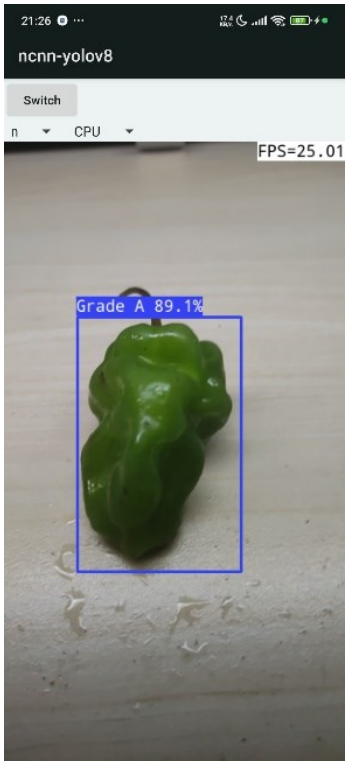
*Source Code* Aplikasi Android: https://github.com/williamadamm/Katokkon-Classification/tree/master/programs/Android/ncnn-android-yolov8

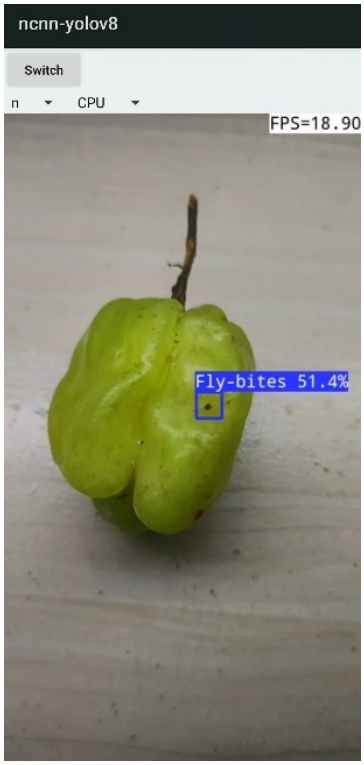
Lampiran 7 Dokumentasi Pengujian

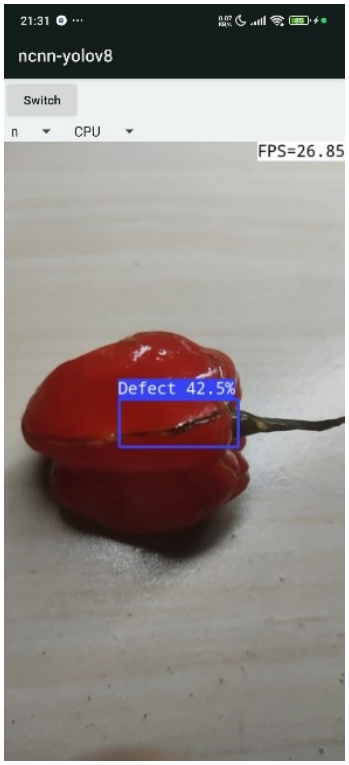



Videos
Chromium Web Browser
VPI Demos v1.2
Jetson Developer Zone
Jetson Support Forums
NVIDIA Jetson Communi...
Jetson Zoo
L4T README
Terminal
0.mp4
0.mp4


192.168.167.97 (raspberrypi) - RealVNC Viewer
pi@raspberrypi: ~/o...
Thonny - /home/pi...
yolov8
19:08
yolov8
2.37
Grade—B 0.69
(x=1023, y=314) ~ R:68 G:73 B:71
Local Python 3 • /usr/bin/python3


21:26
ncnn-yolov8
Switch
n
CPU
FPS=25.01
Grade A 89.1%


ncnn-yolov8
Switch
n
CPU
FPS=18.90
Fly-bites 51.4%


21:31
ncnn-yolov8
Switch
n
CPU
FPS=26.85
Defect 42.5%


ncnn-yolov8
Switch
n
CPU
FPS=7.62
Grade B 82.2%