DAFTAR PUSTAKA

Bahuleyan, H. (2018). *Music Genre Classification using Machine Learning Techniques*. https://doi.org/10.48550/arxiv.1804.01149

Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). *Soft-NMS -- Improving Object Detection With One Line of Code*. https://arxiv.org/abs/1704.04503

Cao, C., Wang, B., Zhang, W., Zeng, X., Yan, X., Feng, Z., Liu, Y., & Wu, Z. (2019). An Improved Faster R-CNN for Small Object Detection. *IEEE Access*, *7*, 106838–106846. https://doi.org/10.1109/ACCESS.2019.2932731

Doshi, S. (2019, January 13). *Various Optimization Algorithms For Training Neural Network*. Towards Data Science. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Emanuella, C. T., Lawi, A., & Hendra. (2022). *Deployment Model Prediksi Harga Saham Apple Inc Pada Beberapa Bursa Efek Menggunakan Metode Multivariate Gated Recurrent Unit* [Universitas Hasanuddin]. http://repository.unhas.ac.id/id/eprint/19091/

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). Fundamental Concepts of Convolutional Neural Network. In V. E. Balas, R. Kumar, & R. Srivastava (Eds.), *Recent Trends and Advances in Artificial Intelligence and Internet of Things* (pp. 519–567). Springer International Publishing. https://doi.org/10.1007/978-3-030-32644-9_36

Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem Dan Teknologi Informasi Indonesia)*, *3*(2), 49–56. https://doi.org/10.32528/JUSTINDO.V3I2.2254

Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Computer Science*, *132*, 679–688. https://doi.org/10.1016/J.PROCS.2018.05.069

Junos, M. H., Mohd Khairuddin, A. S., Thannirmalai, S., & Dahari, M. (2022). Automatic detection of oil palm fruits from UAV images using an improved

YOLO model. *The Visual Computer*, *38*(7), 2341–2355. https://doi.org/10.1007/s00371-021-02116-3

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, *6*(4), 312–315. https://doi.org/10.1016/J.ICTE.2020.04.010

Katole, A. L., Yellapragada, K. P., Bedi, A. K., Kalra, S. S., & Chaitanya, M. S. (2015). *Hierarchical Deep Learning Architecture For 10K Objects Classification*. https://doi.org/10.5121/csit.2015.51408

Kementerian Pertanian, D. J. P. (2022). Statistik Perkebunan Nasional 2020-2022. In *Statistik Perkebunan Indonesia*. Sekretariat Direktorat Jenderal Perkebunan, Kementerian Pertanian.

Latha, R. S., Sreekanth, G. R., Rajadevi, R., Nivetha, S. K., Kumar, K. A., Akash, V., Bhuvanesh, S., & Anbarasu, P. (2022). Fruits and Vegetables Recognition using YOLO. *2022 International Conference on Computer Communication and Informatics (ICCCI)*, 1–6. https://doi.org/10.1109/ICCCI54379.2022.9740820

Lubis, A. U. (2008). *Kelapa Sawit (Elaeis guineensis Jacq.) Di Indonesia* (2nd ed.). Pusat Penelitian Kelapa Sawit.

Nielsen, L. (2021, January 19). *Understanding Torchvision Functionalities (for PyTorch)-Part 1*. The Startup | Medium. https://medium.com/swlh/understanding-torchvision-functionalities-for-pytorch-391273299dc9

Nirthika, R., Manivannan, S., Ramanan, A., & Wang, R. (2022). Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study. *Neural Computing and Applications*, *34*(7), 5321–5347. https://doi.org/10.1007/s00521-022-06953-8

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. https://doi.org/10.48550/arxiv.1811.03378

Padilla, R., Netto, S. L., & Da Silva, E. A. B. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. *International Conference on Systems, Signals, and Image Processing*, *2020-July*, 237–242.

https://doi.org/10.1109/IWSSIP48289.2020.9145130

Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., & Da Silva, E. A. B. (2021). A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, *10*(3), 279. https://doi.org/10.3390/ELECTRONICS10030279

Parico, A. I. B., & Ahamed, T. (2021). Real time pear fruit detection and counting using yolov4 models and deep sort. *Sensors*, *21*(14), 1–32. https://doi.org/10.3390/s21144803

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Sitepu, A. C., & Sigiro, M. (2021). Analisis Fungsi Aktivasi Relu Dan Sigmoid Menggunakan Optimizer Sgd Dengan Representasi Mse Pada Model Backpropagation. *JUTISAL Jurnal Teknik Informatika Universal*, *1*(1), 12–25. https://jurnal.universal.ac.id/index.php/jutisal/article/view/2

Song, Y., Pan, Q. K., Gao, L., & Zhang, B. (2019). Improved non-maximum suppression for object detection using harmony search algorithm. *Applied Soft Computing*, *81*, 105478. https://doi.org/10.1016/J.ASOC.2019.05.005

Supriyono, J. (2017). *Sejarah Kelapa Sawit Indonesia - Gabungan Pengusaha Kelapa Sawit Indonesia (GAPKI)*. https://gapki.id/news/3652/video-sejarah-kelapa-sawit-indonesia

Tzutalin. (2015). *LabelImg*. Git Code. https://github.com/tzutalin/labelImg

Vasilev, I., Slater, D., Spacagna, G., Roelants, P., & Zocca, V. (2019). *Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, 2nd Edition*. Packt Publishing. https://books.google.co.id/books?id=ESKEDwAAQBAJ

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*. 1–15. http://arxiv.org/abs/2207.02696

Zainuddin, Z., Tuluran, J., Achmad, A., Areni, I. S., & Tahir, Z. (2022). The Waste Detection System of Shrimp Feeding with a Waterproof Camera using Yolo

Algorithm. *Journal of Physics: Conference Series*, *2312*(1). https://doi.org/10.1088/1742-6596/2312/1/012083

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2018). Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232. https://doi.org/10.48550/arxiv.1807.05511

Zulkiflie, M. A. (2021). *Implementasi Algoritma Object Detection Yolov4 Dan Euclidean Distance Dalam Mendeteksi Pelanggaran Social Distancing* [Universitas Hasanuddin]. http://repository.unhas.ac.id/id/eprint/12039/

LAMPIRAN

Kode sumber *backend* aplikasi *desktop* untuk menjalankan algoritma YOLOv7 dan melakukan penghitungan:

- Impor pustaka

```
import cv2
import numpy as np
from hubconf import custom
from datetime import datetime
import hashlib

from PyQt5.QtWidgets import QApplication, QMainWindow, QDialog
from PyQt5.QtGui import QImage, QPixmap
from PyQt5 import QtWidgets
from ui.login import Ui_Dialog
from ui.countingScreen import Ui_MainWindow
from ui.database import Database
```

- *Class* halaman *log in*

```
class LoginDialog(QDialog):
    def __init__(self):
        super(LoginDialog, self).__init__()
        self.db = Database()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.loginButton.clicked.connect(self.login)

    def login(self):
        self.username = self.ui.username.text()
        password = self.ui.password.text()
        hashed_password =
self.db.get_hashed_password(self.username)
        hashed_input_password =
hashlib.sha256(password.encode('utf-8')).hexdigest()
        if hashed_password and hashed_input_password ==
hashed_password:
            self.db.last_login(self.username)
            self.accept()
        else:
            QtWidgets.QMessageBox.warning(self, u"Warning",
u"Username atau password salah",
buttons=QtWidgets.QMessageBox.Ok,
defaultButton=QtWidgets.QMessageBox.Ok)
```

- Class halaman utama

```
class Ui_MainWindow(QMainWindow, Ui_MainWindow, object):
    def __init__(self, username):
        super(Ui_MainWindow, self).__init__()
        self.setupUi(self)
        self.db = Database()
        self.image = None
        self.rightCounter = {0: 0}
        self.isPlay = False
        self.yHeight = 40
        self.model = 'models/yolov7-tiny.pt'
        self.source = '0'
        self.username = username

    def getData(self):
        result = self.db.get_data(self.idPersil.text())
        if result:
            tgl_register = result[0][0]
            id_pohon = result[0][1]
            no_pohon = result[0][2]
            barang = result[0][3]
            self.tglRegisterLineEdit.setText(str(tgl_register))
            self.iDPohonLineEdit.setText(id_pohon)
            self.noPohonLineEdit.setText(no_pohon)
            self.varietasLineEdit.setText(barang)
            self.startButton.show()
            self.resetButton.show()
        else:
            QtWidgets.QMessageBox.warning(self, u"Warning",
u"Data tidak ditemukan", buttons=QtWidgets.QMessageBox.Ok,
defaultButton=QtWidgets.QMessageBox.Ok)
```

```
    def updateCounter(self):
        self.countNumber.display(sum(self.rightCounter.values()))

    def resetCounting(self):
        self.rightCounter = {0: 0}
        self.updateCounter()

    def insert_counting(self):
        idpersil = self.idPersil.text()
        count = sum(self.rightCounter.values())
        tgl_register = self.tglRegisterLineEdit.text()
        tgl_counting = datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
        operator = self.username
        self.db.insert_counting(idpersil, count, tgl_register,
tgl_counting, operator)

    def playstop(self):
        if self.isPlay == False:
            self.isPlay = True
            self.startButton.setText("STOP")
            self.startButton.setStyleSheet("background-
color:rgb(255, 0, 0); color:rgb(255, 255, 255)")
            self.startYolo()
        else:
            self.isPlay = False
            self.insert_counting()
            self.startButton.setText("START")
            self.startButton.setStyleSheet("background-
color:rgb(85, 255, 0); color:rgb(255, 255, 255)")
```

```
    def startYolo(self):
        cap = cv2.VideoCapture(self.source)
        model = custom(self.model)
        model.classes = [0]
        model.conf = 0.5
        model.iou = 0.5
        if (cap.isOpened() == False):
            print("Error opening video stream or file")
        ListBBox = []
        while(cap.isOpened() and self.isPlay):
            QApplication.processEvents()
            ret, self.image = cap.read()
            if ret:
                b, c, w = self.image.shape
                MidLineY = b - b*self.yHeight/100
                BBox = self.objectDetection(model, self.image)
                ListBBox.append(BBox)
                if len(ListBBox) > 2:
                    ListBBox.pop(0)
                ListBBox = self.objectTracking(ListBBox)
                self.objectCounting(ListBBox, MidLineY)
                self.image = self.drawLastBoundingBox(self.image,
ListBBox)
                self.image = self.drawObjectVector(self.image,
ListBBox)
                p1 = (0, int(MidLineY))
                p2 = (c, int(MidLineY))
                self.image = cv2.line(self.image, p1, p2, (0,
255, 0), 2)
                self.updateFrame()
            else:
                self.isPlay = False
                self.insert_counting()
                self.startButton.setText("START")
                self.startButton.setStyleSheet("background-
color:rgb(85, 255, 0); color:rgb(255, 255, 255)")
                break
        cap.release()
```

```
    def updateFrame(self):
        frame = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
        image = QImage(frame, frame.shape[1], frame.shape[0],
frame.strides[0], QImage.Format_RGB888)
        self.countingScreen.setPixmap(QPixmap.fromImage(image))
        self.countingScreen.setScaledContents(True)
        self.countingScreen.setScaledContents(True)
```

```
    def objectDetection(self, model, frame):
        results = model(frame)
        resBBox = results.pandas().xyxy[0]
        DetectedBBox = resBBox.values.tolist()
        BBox = []
        pmin = np.array([-1, -1])
        pmax = np.array([-1, -1])
        pmid = (pmin+pmax)/2
        BBox.append([pmin, pmax, pmid, -1, -1, -1, -1])
        for xmin, ymin, xmax, ymax, conf, cl, nama in
DetectedBBox:
            IdPrev = -1
            xmid = (xmin+xmax)/2
            ymid = (ymin+ymax)/2
            BBox.append([xmin, ymin, xmax, ymax, xmid, ymid,
conf, cl, nama, IdPrev])
        return BBox

    def objectTracking(self, ListBBox):
        if len(ListBBox) >= 2:
            CurrentBBox = ListBBox[-1]
            PrevBBox = ListBBox[-2]
            for IndexLast in range(1, len(CurrentBBox)):
                xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc,
clc, namac, IdPrevC = CurrentBBox[IndexLast]
                rCocok = 100000000000
                IndexCocok = -1
                for IndexPrev in range(1, len(PrevBBox)):
                    xminp, yminp, xmaxp, ymaxp, xmidp,
ymidp,  confp, clp, namep, IdPrevp = PrevBBox[IndexPrev]
                    v = np.array([xmidc-xmidp, ymidc-ymidp])
                    RTot = np.linalg.norm(v)
                    if IndexCocok == -1:
                        rCocok = RTot
                        IndexCocok = IndexPrev
                    else:
                        if RTot < rCocok:
                            rCocok = RTot
                            IndexCocok = IndexPrev
                if IndexCocok > -1:
                    ListBBox[-1][IndexLast][9] = IndexCocok
        return ListBBox
```

```
    def objectCounting(self, ListBBox, MidLineY):
        if len(ListBBox) >= 2:
            CurrentBBox = ListBBox[-1]
            PrevBBox = ListBBox[-2]
            for IndexLast in range(1, len(CurrentBBox)):
                xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc,
clc, namac, IdPrevC = CurrentBBox[IndexLast]
                if IdPrevC > -1:
                    xminm, yminm, xmaxm, ymaxm, xmidm, ymidm =
PrevBBox[IdPrevC][0:6]
                    LewatBatas = (ymidc-MidLineY)*(ymidm-
MidLineY)
                    if LewatBatas <= 0:
                        Arah = ymidc-MidLineY
                        if Arah >= 0:
                            self.rightCounter[clc] += 1
                        else:
                            self.rightCounter
            self.updateCounter()

    def drawLastBoundingBox(self, frame, ListBBox):
        CurrentBBox = ListBBox[-1]
        for IndexLast in range(1, len(CurrentBBox)):
            xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc,
clc, namac, IdPrevC = CurrentBBox[IndexLast]
            pc1 = (int(xminc), int(yminc))
            pc2 = (int(xmaxc), int(ymaxc))
            pcc = (int(xmidc), int(ymidc))
            frame = cv2.rectangle(frame, pc1,pc2,(255,0, 255), 1)
            frame = cv2.circle(frame, pcc, 2, (255, 255, 255), 1)
        return frame

    def drawObjectVector(self, frame, ListBBox):
        if len(ListBBox) >= 2:
            CurrentBBox = ListBBox[-1]
            PrevBBox = ListBBox[-2]
            for IndexLast in range(1, len(CurrentBBox)):
                xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc,
clc, namac, IdPrevC = CurrentBBox[IndexLast]
                if IdPrevC > -1:
                    xminm, yminm, xmaxm, ymaxm, xmidm, ymidm =
PrevBBox[IdPrevC][0:6]
                    p1 = (int(xmidc), int(ymidc))
                    p2 = (int(2*xmidc-xmidm), int(2*ymidc-ymidm))
                    frame = cv2.line(frame, p1, p2, (255, 255,
255), 1)
        return frame
```

```
if __name__ == "__main__":
    import sys
    app = QApplication(sys.argv)
    login_dialog = LoginDialog()
    if login_dialog.exec_() == QDialog.Accepted:
        MainWindow = QMainWindow()
        ui = Ui_MainWindow(login_dialog.username)
        ui.setupUi(MainWindow)
        MainWindow.show()
        sys.exit(app.exec_())
    else:
        sys.exit(0)
```